View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by University of Tennessee, Knoxville: Trace

University of Tennessee, Knoxville
Trace: Tennessee Research and Creative Exchange

Masters Theses

Graduate School

5-2008

A New Fully Implicit in Time Two-Dimensional Inverse Heat Conduction Method

Bryan S. Elkins
University of Tennessee - Knoxville

Recommended Citation
Elkins, Bryan S., "A New Fully Implicit in Time Two-Dimensional Inverse Heat Conduction Method. " Master's Thesis, University of Tennessee, 2008.
https://trace.tennessee.edu/utk_gradthes/355

This Thesis is brought to you for free and open access by the Graduate School at Trace: Tennessee Research and Creative Exchange. It has been accepted for inclusion in Masters Theses by an authorized administrator of Trace: Tennessee Research and Creative Exchange. For more information, please contact trace@utk.edu.

To the Graduate Council:

I am submitting herewith a thesis written by Bryan S. Elkins entitled "A New Fully Implicit in Time Two-Dimensional Inverse Heat Conduction Method." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Aerospace Engineering.

Majid Keyhani, Major Professor

We have read this thesis and recommend its acceptance:

Jay Frankel, Rao Arimilli

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Bryan Scott Elkins entitled "A New Fully Implicit in Time Two-Dimensional Inverse Heat Conduction Method." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Aerospace Engineering.

Majid Keyhani

Major Professor

We have read this thesis
and recommend its acceptance:

Jay Frankel

Rao Arimilli

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and

Dean of the Graduate School

(Original signatures are on file with official student records)

A New Implicit in Time with Space Marching Two-Dimensional Inverse Heat Conduction Method

A Thesis Presented for
the Master of Science Degree
The University of Tennessee, Knoxville

Bryan Scott Elkins
May 2008

Acknowledgements

First, I would like to thank my advisors, Drs. Majid Keyhani and Jay I. Frankel, for their support and guidance throughout this thesis work. I am also grateful for the financial support provided by Dr. Frankel through the National Science Foundation and the MABE department. In addition to my advisors, I would also like to thank Dr. Rao V. Arimilli for his comments and help in the preparation of this thesis. I am appreciative of the opportunity to work on this project.

Personally, I would like to thank my parents for their constant support and encouragement. Finally, I would like to thank my wife for listening to my ideas and for the endless encouragement she always provides.

Abstract

A new combination of methods used to solve the transient, two-dimensional inverse heat conduction problem (IHCP) is presented in this thesis. A simple implicit in time with space marching approach is used in combination with digital filtering for regularization. Results are presented for both one-dimensional and two-dimensional problems. As much as 10% measurement error is added to the data to simulate experimental results.

One-dimensional results with "perfect" data suggested that refining the spatial and temporal meshes improved the accuracy of the inverse solution. However, the nodal Fourier number was found to have no effect on accuracy. Further investigation into the effect of the Fourier number on the inverse solution led to the discovery of a precisely defined stability criterion, which is presented for the one-dimensional problem.

The digital filter was proven to be a highly effective regularization method for both the one-dimensional and two-dimensional cases. The Gauss low pass filter employed a cutoff frequency as the regularization parameter, which has an exact definition with physical significance. No trial and error method of choosing a regularization parameter was necessary. Temperature and heat flux data were given at the sensor sites, noise was added to the data, filtered, and used as input data to the inverse code. The prediction error resulting from the use of "perfect" data suggested a bias (over-prediction). With this bias removed from the noisy, filtered results, the inverse solution was accurate to 2% for the 1D case and 3% for the 2D case. The dramatic

reduction of error through an inverse process was striking, since the inverse problem is well-known to magnify measurement errors.

An additional case using only one line of temperature data and no heat flux data was used as alternate approach to the inverse problem. The new strategy relies on a recently developed integral relationship between the heating rate and the heat flux, which is valid on the half-space. The relationship allows for the heat flux to be found at the sensor site using only temperature and heating rate data. This data could be experimentally obtained using one line of thermocouples combined with numerical differentiation to obtain the heating rate; no heat flux gauges or second series of thermocouples are necessary. This thesis is the first to take advantage of heat flux – heating rate relationship. Results from this method with a measurement error of 5% predicted a surface heat flux error of only 5%.

Table of Contents

List of Figures

Nomenclature

a_P	finite difference coefficient, subscripts E, W, S and N also used, W/(m•K)
a_P^0	finite difference storage coefficient, W/(m•K)
b	half-width of the heating element on the surface, m
C	specific heat capacity, J/(kg•K)
d	x-distance from the surface to the line of embedded sensors, m
f_c	cutoff frequency used in digital filter, Hz
Fo	Fourier number, $\alpha t/d^2$
Fo_{cv}	nodal Fourier number, $\alpha \Delta t/\Delta x^2$
Fo_p	Fourier number of penetration, $\alpha t_p/d^2$
$Fo_{\Delta t}$	time step Fourier number, $\alpha \Delta t/d^2$
H	half height of the domain in the y-direction, m
k	thermal conductivity, W/(m•K)
L	width of the domain in the x-direction, m
M	number of computational nodes in the x-direction
m	a certain x-direction line of nodes
N	number of computational nodes in the y-direction
n	a certain y-direction line of nodes
q_o''	pulse heat flux strength at surface, W/m^2
q_x''	heat flux in the x-direction, W/m^2
P	number of time steps used

T	temperature variable, ^{o}C
T_o	initial temperature, ^{o}C
t	time variable, s
t_p	penetration time, s
t_o	dummy time variable, s
x	spatial variable in the x-direction, m
y	spatial variable in the y-direction, m
y_o	dummy spatial variable in the y-direction
α	thermal diffusivity, m^2/s
Δt	time step for the finite difference method, s
Δx	distance between nodes in the x-direction, m
Δy	distance between nodes in the y-direction, m
ε	measurement error, % signal
θ	dimensionless temperature response, $(T - T_o)/(q_o'' d / k)$
ρ	density, kg/m^3

Chapter 1 Introduction

1.1 Problem Description

The determination of an unknown surface heat flux through measurements taken somewhere in the interior of a body is called an inverse heat conduction problem (IHCP). IHCPs are of particular interest when the surface is under a harsh environment, such as high temperatures or high heat fluxes, which precludes the use of sensors on the surface directly. The two-dimensional geometry considered in this thesis can be seen in Figure (1a). A heat source of half width, b, is imposed on the center of the surface. The surface is insulated everywhere else. The length and width of the geometry are chosen so as to prevent the thermal front from reaching the sidewalls or the back wall for the duration of the experiment.

The inverse problem is unstable. That is to say a small amount of error in the measurements produces a large error in the surface prediction [1]. Temperature and heat flux measurements are taken at the sensor site below the surface (x = d), which in the presence of noise will produce unstable surface predictions. Investigators often employ a "regularization parameter" to eliminate this instability, but the choice of regularization parameter is somewhat arbitrary with little physical relationship to the problem at hand [1,2].

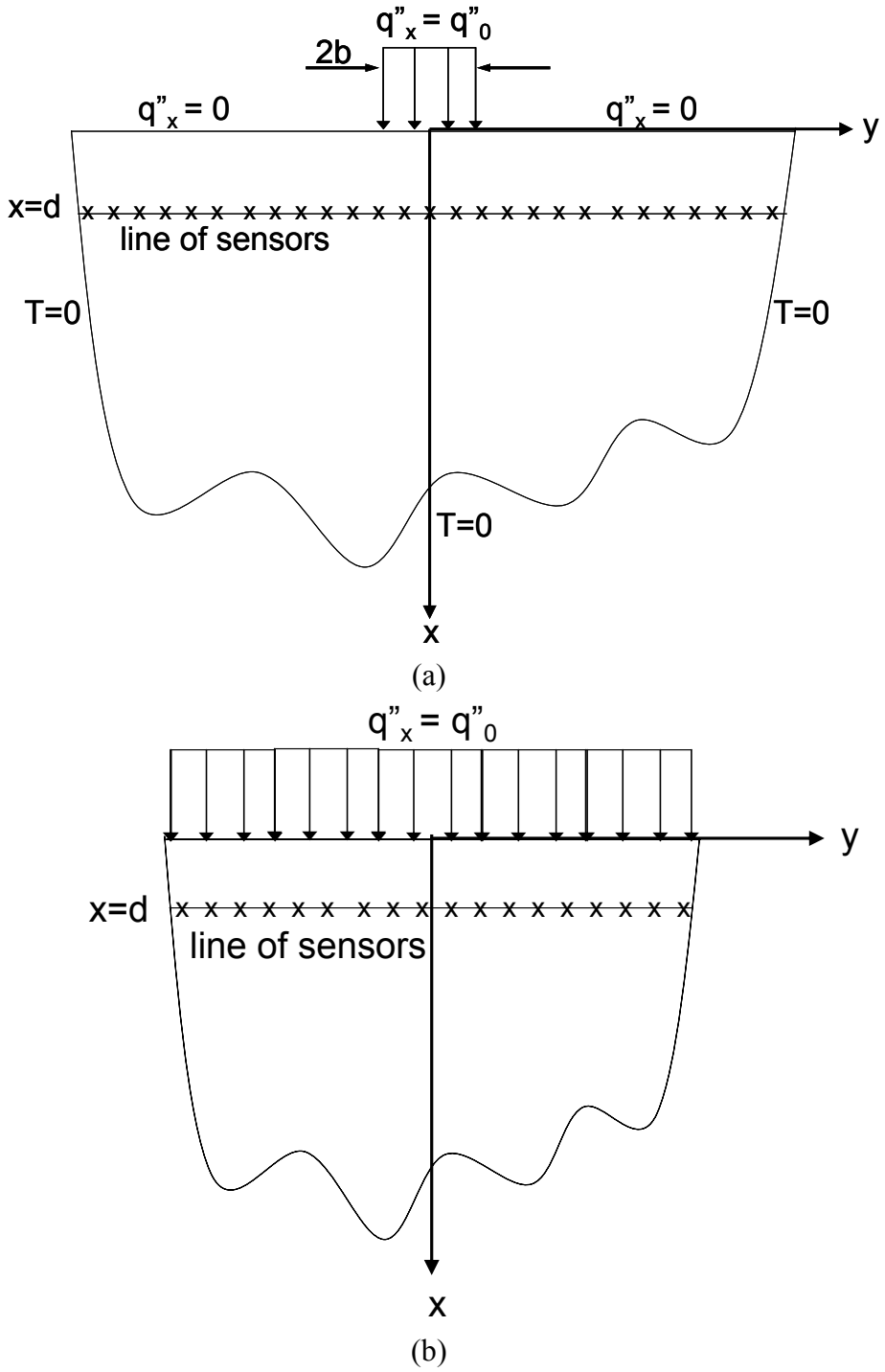


Figure 1: Geometry for analytic problem. a) Two-dimensional half space and b) One-dimensional half space.

1.2 One-Dimensional Problem

If the surface heat flux of Figure (1a) is extended all the way across the surface and the sidewalls are insulated, the problem becomes one-dimensional. This geometry can be seen in Figure (1b). The governing equation for this domain is the transient one-dimensional heat conduction equation with constant properties which is given by

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}, \quad x,t \geq 0 \tag{1a}$$

subject to the initial condition

$$T(x,t=0) = T_0 \tag{1b}$$

where α is the thermal diffusivity. The surface heat flux is given by Fourier's law as

$$q_x''(0,t) = q_x'' = -k \frac{\partial T}{\partial x}(0,t), \quad t > 0 \tag{1c}$$

and the half space boundary condition is given by

$$T(\infty,t) = 0, \quad t \geq 0 \tag{1d}$$

where k is the thermal conductivity. The exact solution to Equations (1a-1c) is given by Incropera and DeWitt's [3, p270] constant surface heat flux solution on the semi-infinite solid as

$$T(x,t) - T_0 = \frac{2q_0''(\alpha t/\pi)^2}{k} \exp\left(\frac{-x^2}{4\alpha t}\right) - \frac{q_0'' x}{k} erfc\left(\frac{x}{2\sqrt{\alpha t}}\right), \quad x,t \geq 0 \tag{2}$$

where erfc is the complementary error function [4].

The unstable nature of the inverse problem is apparent by investigating the relationship between the heat flux and the heating rate, $\partial T/\partial t$, given by [2]

$$q_x''(x,t) = \sqrt{\frac{\rho C k}{\pi}} \int_{u=0}^{t} \frac{\partial T}{\partial u}(x,u) \frac{du}{\sqrt{t-u}}, \quad x,t \geq 0 \tag{3}$$

where ρ is the density and C is the specific heat capacity. Note that Equation (3) is only valid on half space, ie, the thermal front must not reach the back wall for the duration of the experiment. The measured temperature data at a sensor site below the surface (x = d), which can be assumed to have some degree of noise, must be numerically differentiated. This operation only serves to magnify the noise and Equation (3) becomes an ill-posed problem [2]. If, however, the heating rate is directly measured, Equation (3) is a well-posed problem since the integration tends to smooth noisy data. Since there is currently no sensor available to directly measure the heating rate, care must be taken to ensure a smooth heating rate curve. This can be accomplished by careful digital filtering. Frankel and colleagues [2,5] recommends the use of a digital filter to accomplish this and have proven its effectiveness.

1.3 Two-Dimensional Problem

The governing equation for two-dimensional problem seen in Figure (1a) is given by

$$\frac{\partial T}{\partial t}(x,y,t) = \alpha \nabla^2 T(x,y,t), \quad y \in (-\infty, \infty), \quad x,t \geq 0 \tag{4a}$$

subject to boundary conditions

$$T(x,y,0) = 0, \quad x \geq 0, \quad y \in (-\infty, \infty) \tag{4b}$$

$$q_x''(0,y,t) = q_0'' = -k \frac{\partial T}{\partial x}(0,y,t), \quad |y| \leq b, \quad t > 0 \tag{4c}$$

$$q_x''(0,y,t) = -k\frac{\partial T}{\partial t}(0,y,t) = 0, \quad |y| > b, \quad t > 0 \tag{4d}$$

$$T(\infty,y,t) = 0, \quad y \in (-\infty,\infty), \quad t \geq 0 \tag{4e}$$

$$T(x,-\infty,t) = T(x,\infty,t) = 0, \quad x,t \geq 0 \tag{4f}$$

Similar to the one-dimensional case, a relationship exists between the heat flux and the heating rate on the half space is given by [5]

$$q_x''(x,y,t) = \frac{k}{2\alpha\pi}\int_{t_o=0}^{t}\int_{y_o=-\infty}^{\infty}\frac{e^{-\frac{(y-y_o)^2}{4\alpha(t-t_o)}}}{(t-t_o)}\left[\frac{\partial T}{\partial t_o}(x,y_o,t_o)\right.$$

$$\left.+\frac{T(x,y_o,t_o)}{2(t-t_o)}\left(1-\frac{(y-y_o)^2}{2\alpha(t-t_o)}\right)\right]dy_o dt_o, \quad x,t \geq 0, \quad y \in (-\infty,\infty) \tag{5}$$

where y_o and t_o are dummy variables for y and t, respectively. Equation (5) is only valid if the thermal front does not reach the sidewalls or back wall for the duration of the experiment. Note Equation (5) involves both the heating rate and temperature while Equation (3) only involves the heating rate. Similar to the one-dimensional formulation of Equation (3), Equation (5) conveys the importance of a smooth heating rate needed for a well-posed integration. This relationship implies that the heat flux at a measurement site could be determined from a single line of thermocouples combined with an effective digital filter, eliminating the need for complex heat flux gauges or a second line of thermocouples to generate a spatial gradient for approximating q_x'' from Fourier's law.

The two-dimensional IHCP is an area which is not as extensively investigated as the one-dimensional version. This is due to the complex and unstable nature of the problem. A space marching finite difference scheme is a commonly used method to

solve the two-dimensional IHCP. However, these methods require the use of a regularization parameter to overcome the ill-posed nature of inverse problems. To this end, some one and two-dimensional methods commonly employ a complex high order space-marching scheme [6,7] where the effects of neighboring nodes are damped out to eliminate instability. Tikhonov regularization is often used and is similar to filtering, but the parameter is somewhat arbitrarily chosen through a trial and error method [1].

1.4 Purpose and Organization of Thesis

This thesis attempts to improve current two-dimensional (2D) inverse analysis and presents a new IHCP solution method with specific stability criteria and a physically derived regularization parameter. No knowledge of the surface condition is known a priori. The approach of Frankel and colleagues presented above will be taken advantage of together with a digital filter to regularize the data and obtain a smooth heating rate curve. Whereas traditional inverse methods require either two series of embedded thermocouple temperature data parallel to the surface, or a combination of temperature and heat flux data, Equation (3) will be exploited enabling the IHCP to be solved using only one series of thermocouples. Contrary to the complex space marching formulation, a straightforward fully-implicit in time space with marching scheme will be employed, further reducing instability and allowing for a definite stability criteria to be established.

The new IHCP method is first investigated for the one-dimensional (1D) case. Perfect data (temperature and heat flux) supplied beneath the surface are used to predict surface temperature and heat flux. This first step is used to validate the code and explore the characteristics of the method itself. Up to 10% white noise is then added to simulate

real data and filtered with good inverse results. Finally, Equation (2) is employed with only temperature data given to verify the entirety of the 1D method, and spatial and temporal meshing effects are explored.

The two-dimensional IHCP is next explored in the same order as the 1D case. Perfect temperature and heat flux data from in-situ measurements are first used to predict the surface temperature and heat flux histories and surface spatial distributions, verifying the accuracy of the code. Up to 10% white noise is again added and filtered to simulate a real experiment. The heat flux – heating rate relationship of Equation (3) is then employed with one line of "thermocouples" parallel to the surface to show the effectiveness of the method. Detailed effects of both spatial and temporal mesh are investigated.

As stability is of paramount concern for stable and accurate resolution of the inverse problem, a stability criterion is investigated as well. A simple rule involving penetration time, sensor depth, and time step size will be shown providing unconditional stability.

Chapter 2 Literature Review

Jacques Hadamard was the first to bring about the issue of a well-posed problem [1,8]. According to Hadamard, a problem required existence, uniqueness, and stability of solutions [1] to be well-posed. Any problem which did not meet this criteria was considered to be ill-posed. The IHCP has since been proven to be one such ill-posed problem and studied extensively.

Beck [1] describes several one-dimensional IHCP solution methods. The key methods outlined include exact solutions, regularization, and space marching methods. The discussion of one-dimensional methods presented herein draws heavily from Beck.

The method of Burgraff [9] using an exact solution for the transient one-dimensional IHCP assumes a known temperature and heat flux boundary condition at the sensor site. It uses an infinite series to solve for the surface heat flux. The negative aspect to this scheme is that it requires all-order derivatives of both temperature and heat flux data which is not practical for a real experiment. The infinite series is commonly truncated to a few terms (N, for instance) which still requires N derivatives of discrete noisy data, making this an unstable solution method, and must be regularized in some manner.

Regularization methods can vary in implementation as either whole domain (Tikhonov regularization) or sequential schemes. These procedures are closely akin to forms of least squares known as “damped least squares” [1] and methods for solving nonlinear least squares problems. The goal of the regularization method is to minimize

the sum of the squares function, S, which damps out high frequency changes in the sensor temperature and heat flux data [10]. This is accomplished by choice of a "regularization parameter" which is obtained by the residual sum of squares equal to the numerical value expected. Tikhonov suggests accomplishing this through a trial-and-error method. However, once a regularization parameter is established for a specific problem type, it can be applied to problems of a similar nature.

One-dimensional space marching methods use either an implicit or explicit temporal formulation [1,6,7]. Space marching is essentially a finite difference scheme where one spatial node can be solved for directly using the known conditions. This node's solution is then used to solve the next node and so on. Two temperature sensors are commonly assumed as the known boundary data. The simplest implicit formulation, the space marching method is very susceptible to measurement error. Various schemes have been used to damp out any noise in the data, which involve the use of temperature data from future times [1].

The two-dimensional IHCP has also been explored, though not as extensively as the one-dimensional case. Methods that have been used, however, include finite difference in combination with Laplace transform sequentially in-time [11] and space-marching methods [12].

Chen et al. [11] present a two-dimensional inverse method employing a Laplace transform technique and finite differencing sequentially in time. In order to facilitate the Laplace transform, they use a polynomial fit of the data and a least squares minimization technique to reduce the effects of measurement error. Good results are obtained for noisy data, however, the more difficult prediction of surface heat flux is not presented.

Taler and Zima [12] presented a two-dimensional inverse solution method utilizing a simple space-marching scheme. Their geometry was a rectangular bar with convection on two sides (north and east), each side with a different heat transfer coefficient. The other two sides were insulated. Heat flux and temperature data were given at the sensor locations. To eliminate noise in the data, they used a Gram-Schmidt orthogonal piecewise cubic polynomial and a least squares procedure [12]. Good results were obtained, however the method presented only included a total of 15 nodes. This enabled the surface heat flux prediction at only 1 node. This is acceptable for their test case, as the entire surface was experiencing the same heat flux, but it would prove ineffective at reproducing a distributed surface heat flux.

Additional advancements in inverse research have been made by Kulish et al. [13-14], and Frankel [2,5]. Kulish derived a new relationship between temperature and heat flux one dimension using Laplace transforms. Frankel et al. [2,5] was able to generalize the method using either Green's functions or Fourier transforms to obtain the relationship in two-dimensions with potential extension to three dimensions [5]. As heat flux rather than temperature is of interest, Frankel et al. also inverted the temperature – heat flux relationship of Kulish to obtain a heat flux – heating rate relationship. This relationship enables the use of a single line of thermocouples in the y-direction to predict the sensor heat flux in the x-direction. However, no researcher has yet implemented this idea into a two-dimensional inverse problem.

It is the goal of this study to use the best of each method. Regularization methods produce good results, but the trial and error choice of parameter should be avoided if possible. Space marching methods in their simplest form are easy to implement but need

regularization. Therefore, the proposed method will use the easy to implement space marching and a filtering scheme which has a similar effect to regularization but with a parameter that has a strong physical meaning.

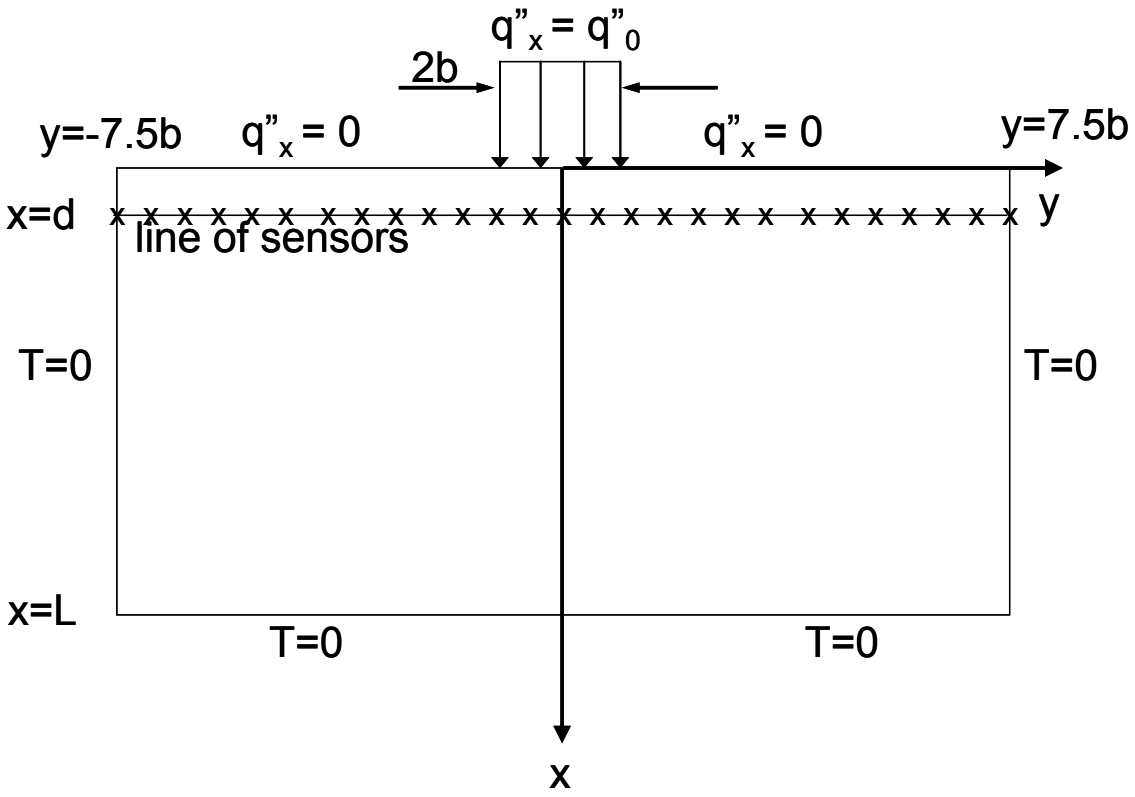
Chapter 3 The Inverse Method

The new IHCP solution method presented is for transient conduction in a two-dimensional geometry seen in Figure (1a). This thesis employs the new heat flux – heating rate relationship developed by Frankel et al. [5] with only one line of temperature sensors to obtain the sensor heat flux. Many of the present inverse methods available to solve such a problem employ a complex regularization scheme as noted in the Literature Review. In contrast, a straightforward Gauss low-pass filter will be used to eliminate high frequency noise.

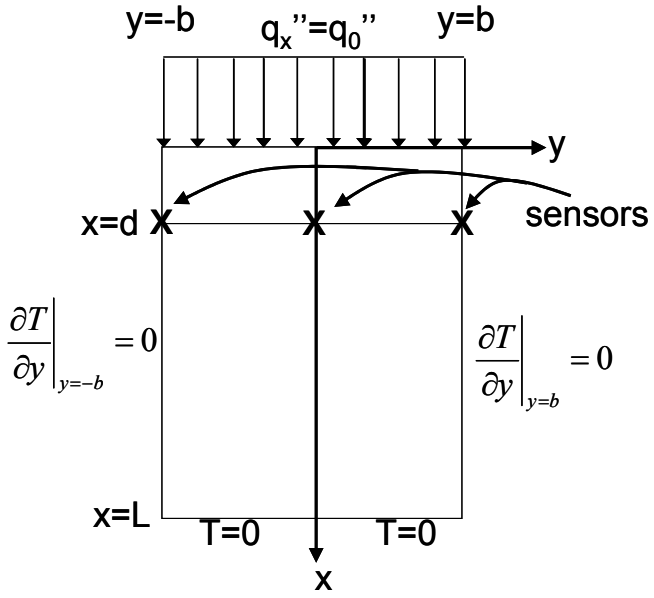
Since no experimental data was available for this problem, a computer simulation was used. First, a direct finite difference code was used to obtain the "perfect" temperature data located a certain depth, d, which will be used as the boundary condition data at the southern boundary of the inverse problem. Second, noise was added to the "perfect" data to simulate experimental data. Third, a digital filter was employed to remove the unwanted noise. Fourth, the heat flux – heating rate relationship was utilized to obtain the heat flux at the sensor sites. Lastly, a fully-implicit in time finite difference method with space marching was used to solve for the unknown boundary condition.

3.1 Direct Method

The half-space geometries of Figure (1) were transformed into a domain with finite boundaries shown in Figure (2) such that the half-space conditions existed for the duration of the experiment. After applying Patankar's discretization scheme [15] to the



(a)



(b)

Figure 2: Problem geometry for the study. (a) The general two-dimensional case and (b) the one-dimensional case.

geometry of Figure (2), finite difference equations were solved via the explicit method. Temperature and heat flux data were export at the x = d sensor line for use as boundary data for the inverse problem. The explicit method was used as it more accurately represents the penetration time, t_p; a finite amount of time is required for control volumes located a depth below the surface to "feel" the heat flux at the surface. The exact solution for the one-dimensional problem in Equation (2) was used to define the penetration time. Equation (2) was made dimensionless using the dimensionless temperature response, θ, and the Fourier number.

$$\theta(Fo) = \frac{2}{\sqrt{\pi}}\sqrt{Fo}\exp\left(\frac{-1}{4Fo}\right) - erfc\left(\frac{1}{2\sqrt{Fo}}\right) \tag{6a}$$

$$\theta = \frac{T(x,t) - T_0}{q_0'' d/k} \tag{6b}$$

$$Fo = \frac{\alpha t}{d^2} \tag{6c}$$

The Fourier number of penetration Fo_p was defined as the Fo when θ reached 0.01. From Figure (3), it can be seen that $Fo_p = 0.13$.

3.2 Inverse Solution Method

A straightforward implicit in time with space marching approach was used to solve the inverse problem. The method used a regularization parameter with a physical meaning and assumed no knowledge of the solution a priori. The discretization scheme for the inverse method can be seen in Figure (3), using a zero control volume at the line of sensors, and a half control volume at the surface.

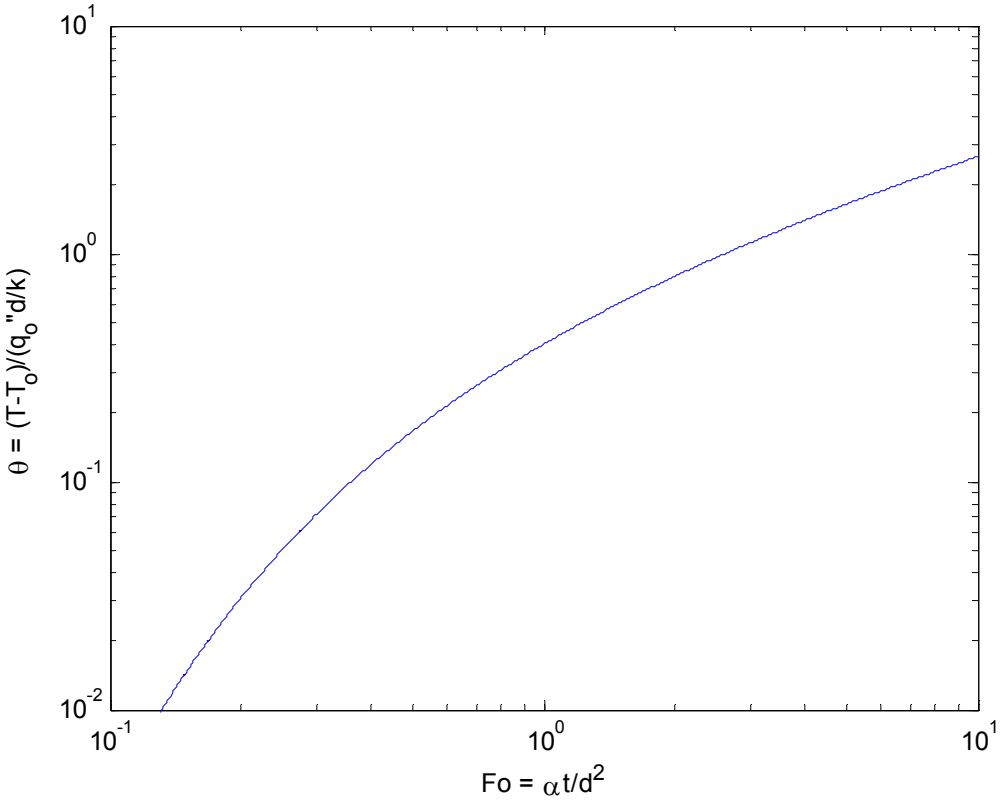


Figure 3: Dimensionless temperature response, θ, to a constant surface heat flux, $q_o^{"}$, at x = d below the surface to a constant surface heat flux. The Fourier number of penetration, Fo_p, defined as the Fo when θ = 0.01, is seen to be 0.13.

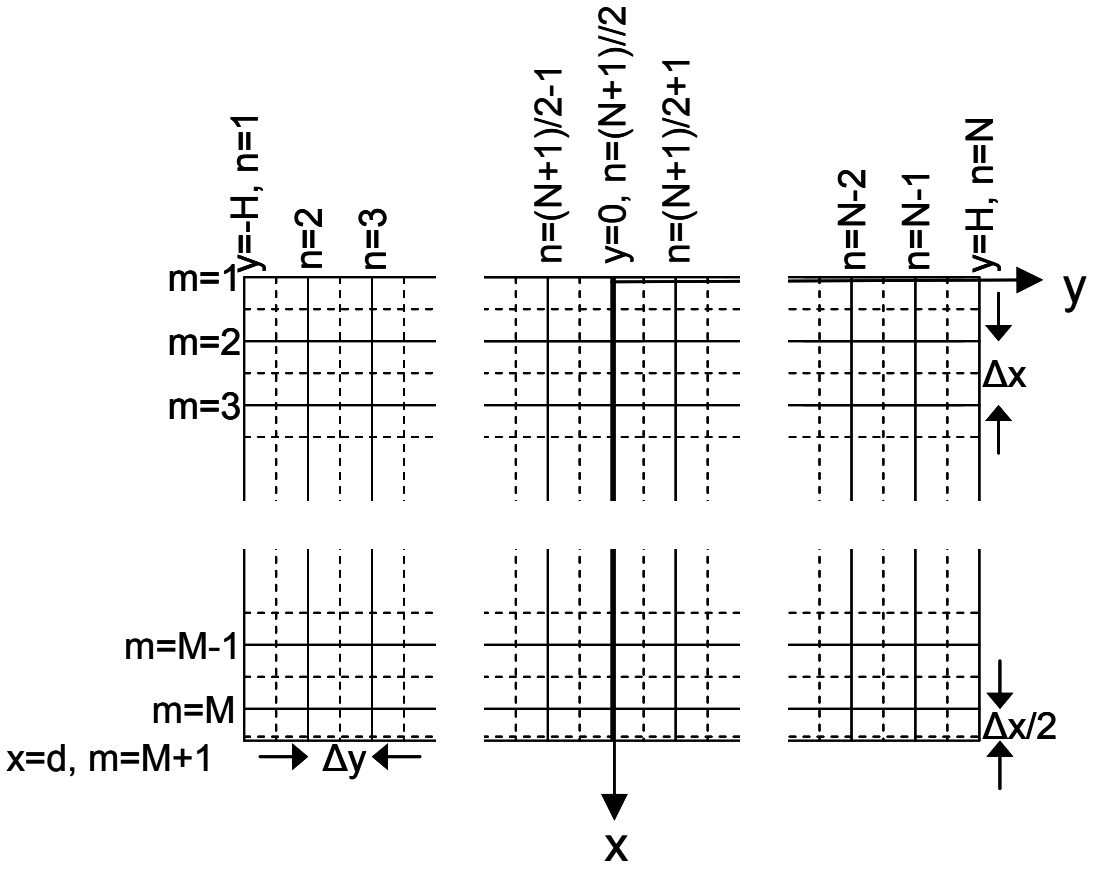


Figure 4: Discretization scheme used for two-dimensional boundary condition problem.

The inverse method was solved in a straightforward manner. For clarity, the notation, $T_{m,n}$, where m and n correspond to the x and y location as seen in Figure (4) will be used in this discussion. Since the a zero control volume was assumed at the sensor location, m=M=1, the temperature solution for the m=M row above the line of sensors was given explicitly through Fourier's Law in terms of known data:

$$T_{M,n} = T_{B,n} + \frac{q''_{B,n}\Delta x}{2k} \tag{7}$$

where T_B and q''_B are the sensor data. Note that Equation (7) only involves the temperature and heat flux sensor data. Now using a fully implicit in time finite difference formulation, all subsequent rows of data, 1 < m < M, were explicitly given by

$$T_{m-1,n} = \frac{1}{a_N}\left[(a_P T)_{m,n} - (a_E T)_{m,n+1} - (a_W T)_{m,n-1} - (a_S T)_{m+1,n} - (a_P^0 T^0)_{m,n}\right] \tag{8a}$$

and all "a" coefficients are given by Patankar's method [12] as

$$a_E = a_W = \frac{k\Delta y}{(\delta x)_e} = \frac{k\Delta y}{\Delta x} \tag{8b}$$

$$a_S = a_N = \frac{k\Delta x}{(\delta y)_s} = \frac{k\Delta x}{\Delta y} \tag{8b}$$

$$a_P^0 = \frac{\rho C \Delta x \Delta y}{\Delta t} \tag{8c}$$

$$a_P = a_E + a_W + a_S + a_N + a_P^0 \tag{8d}$$

Beginning at m=M and progressing to the surface, m=2, enabled a straightforward solution where there was always only one equation and one unknown. No future times were used in the formulation as the digital filter was employed to remove any unwanted noise. The surface heat flux was then given by

$$q''_{1,n} = \frac{1}{\Delta y}\left[(a_P^* T)_{1,n} - (a_E^* T)_{1,n+1} - (a_W^* T)_{1,n-1} - (a_S T)_{2,n} - (a_P^{*0} T^0)_{1,n}\right] \tag{9a}$$

where

$$a_E^* = a_W^* = \frac{k\Delta y}{\Delta x / 2} \tag{9b}$$

$$a_P^{*0} = \frac{\rho C_p (\Delta x / 2)\Delta y}{\Delta t} \tag{9c}$$

$$a_P^* = a_E^* + a_W^* + a_S + a_P^{*0} \tag{9d}$$

3.3 Digital filtering of data

As with all space marching methods, this inverse method will blow up when noisy data is used. Frankel has shown that the use of a Gauss low-pass filter with a specified cut-off frequency, f_c, is highly efficient at removing unwanted noise [2,5]. This works since the nature of conduction is to damp out higher frequencies, thus the only information the thermocouple receives from conduction is of low frequency. Therefore, higher frequencies visible in the Discreet Fourier Transform (DFT) can be filtered out using a low-pass filter with a specified cut-off frequency. The cut-off frequency is the "regularization parameter" for this inverse method which is based on a physical value and can be determined from the DFT. Thus noise will be added to the "perfect" data from the direct problem and filtered using a Gauss low-pass filter given by

$$\Psi_{filter}(x,y,t) = \frac{\sum_{n=0}^{M} \Psi_{noise}(x,y,t_n) e^{-\frac{(t-t_n^2)\varpi_c^2}{4}}}{\sum_{n=0}^{M} e^{-\frac{(t-t_n^2)\varpi_c^2}{4}}}, \quad t \geq 0 \tag{10}$$

where Ψ can be heat flux or temperature, $\omega_c = 2\pi f_c$ and f_c is the cutoff frequency chosen as described in [2,3]. Therefore, using the temperature and heat flux data (direct problem) at the x=d line, T_{filter} and q_{filter} will be determined for use as the x=d boundary condition "data" in the inverse problem. A more detailed discussion of this filtering discussion is given in Appendix A.

3.4 Sensor heat flux determination

Frankel et al. [2,5] have shown the relationship between temperature and heating rate is given by

$$q_x^{"}(x,t) = \sqrt{\frac{\rho C k}{\pi}} \int_{u=0}^{t} \frac{\partial T}{\partial t}(x,u) \frac{du}{\sqrt{t-u}}, \quad x,t \geq 0 \tag{3}$$

for one spatial dimension and by

$$q_x^{"}(x,y,t) = \frac{k}{2\alpha\pi} \int_{t_o=0}^{t} \int_{y_o=-\infty}^{\infty} \frac{e^{-\frac{(y-y_o)^2}{4\alpha(t-t_o)}}}{(t-t_o)} \left[\frac{\partial T}{\partial t_o}(x,y_o,t_o) + \frac{T(x,y_o,t_o)}{2(t-t_o)} \left(1 - \frac{(y-y_o)^2}{2\alpha(t-t_o)} \right) \right] dy_o dt_o \tag{5}$$

$$x,t \geq 0, \quad y \in (-\infty,\infty)$$

for two spatial dimensions. Therefore, Equation (3) was used to predict the sensor heat flux data. Although the relationship is highly singular when $y = y_o$, good results were obtained using simple trapezoidal integration and singularity subtraction. Namely, the expression

$$q_x^{"}(x,y,t) = \frac{k}{2\alpha\pi}\sum_{i=0}^{P} w_i \sum_{j=-N}^{N} v_j \frac{e^{-\frac{(y-y_j)^2}{4\alpha(t-t_i)}}}{(t-t_i)}\left[\frac{\partial T}{\partial t_o}(x,y_j,t_i)\right.$$

$$\left. + \frac{T(x,y_j,t_i)}{2(t-t_i)}\left(1-\frac{(y-y_j)^2}{2\alpha(t-t_i)}\right)\right] \quad x,t \geq 0, \quad y \in (-\infty,\infty) \tag{11a}$$

where v_r and w_i are weights given by

$$w_i = \begin{cases} \frac{\Delta t}{2}, & i = 1, P \\ \Delta t, & i \neq 1, P \end{cases} \qquad v_r = \begin{cases} \frac{\Delta x}{2}, & j = -N, N \\ \Delta x, & j \neq -N, N \end{cases} \tag{11b,c}$$

was evaluated for x=d (sensor location) where P is the number of temporal nodes 2N+1 spatial nodes were used.

3.5 Method Implementation

In order to validate and optimize the two-dimensional inverse code, it was first applied to the one-dimensional case shown in Figure (2b). Since the BC at the north face is constant for all time and across the entire domain, the problem is obviously one-dimensional, meaning that only one node in the y-direction is necessary. However, three nodes in the y-direction were used so as to validate the two-dimensional method outlined in Section 3.2, recognizing that the data reported at TC1, TC2 and TC3 shown in Figure (2b) should be identical to each other.

After successfully implementing the code for a one-dimensional boundary condition, a two-dimensional boundary condition was imposed seen in Figure (2a). A

heat flux varying in the y-direction and constant in time was used. The results from this section prove that the method is valid and highly accurate.

In order to simulate a real experiment, properties of Cr-Ni steel specified at room temperature [16] were used: $\rho = 7900$ kg/m^3, $k = 14.70$ W/(m x K), and $\alpha = 3.75(10^{-6})$ m^2/s. As this method was designed to replicate the physical world, it was assumed that the minimum distance obtainable between sensors was 0.5 cm. This value was used as the spacing in the y-direction, Δy, for the inverse problem. Geometry parameters seen in Figure (1a) were chosen to be $q_0^{"} = 1(10^6)$ W/m^2, b = 1.0 cm, and d = 0.5 cm. Recalling from Figure (2) that the Fourier number of penetration was 0.13, this yields a penetration time of 0.88 seconds for this problem.

Chapter 4 One-Dimensional Results

In order to validate and optimize the two-dimensional inverse code, it was first applied to the one-dimensional case with 3 nodes in the y-direction. This domain can be seen in Figure (2a). The following analysis was performed to determine the optimum number of time steps and nodes in the x direction for accuracy and stability.

4.1 One-Dimensional, Perfect Data

A plot of the resultant inverse-predicted temperature and heat flux can be seen in Figure (5). The inverse solution was accurate to 1% for both temperature and heat flux at the surface. Also note that the data required a finite amount of time to penetrate to the sensors, called the penetration time. For this reason, the inverse code was unable to predict data until after penetration time, and, as such, future history plots will remove solution data prior to the penetration time.

A finite penetration time says that at $t = t_p$, the sensors are just beginning to feel the effects of the heat transfer at surface at t = 0s. This implies that the inverse code at $t = t_p$ could be predicting what happened at the surface at t = 0s, and, should therefore be shifted backwards in time by t_p. However, the results from this numerical example accurately predicted the solution at the surface for unshifted time. That is to say sensor data at $t = t_p$ was used to accurately predict the surface condition at $t = t_p$. Therefore, the solutions presented in this thesis are not shifted backward in time by the penetration time. This effect of penetration time is important, and requires further investigation.

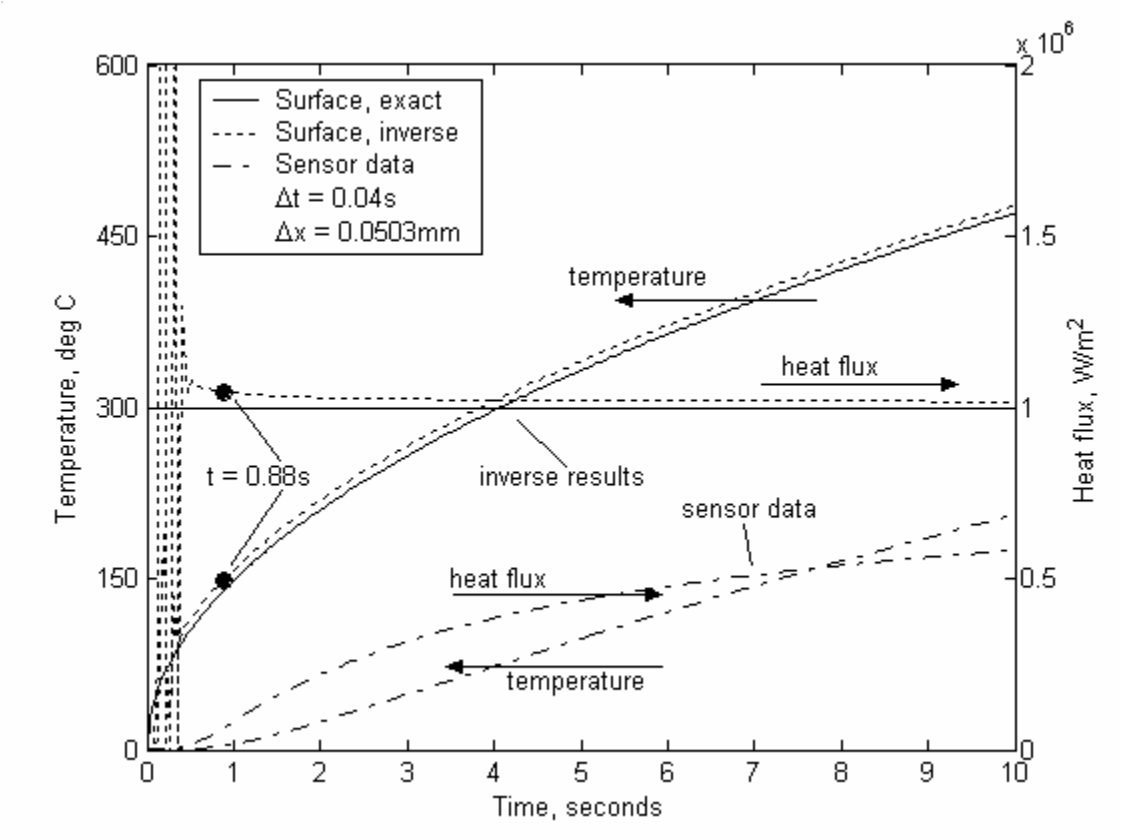


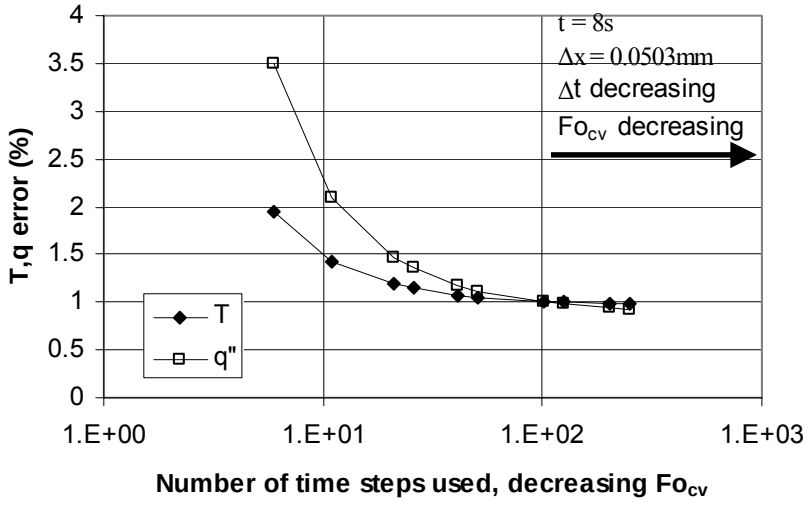
Figure 5: One-dimensional inverse surface temperature and heat flux solution using "perfect" data (ε = 0%) from the direct problem.

The results were found to vary with the chosen time and space mesh. Therefore problem parameters were investigated to discover the optimum mesh. These parameters included Fourier number, spacing in the x-direction, as well as the time step size. Note that in the case of perfect data, all three nodes in the y-direction at a given x-location had identical information for all time. For initial investigations, temperature and heat flux at the sensor site was provided from the direct method.

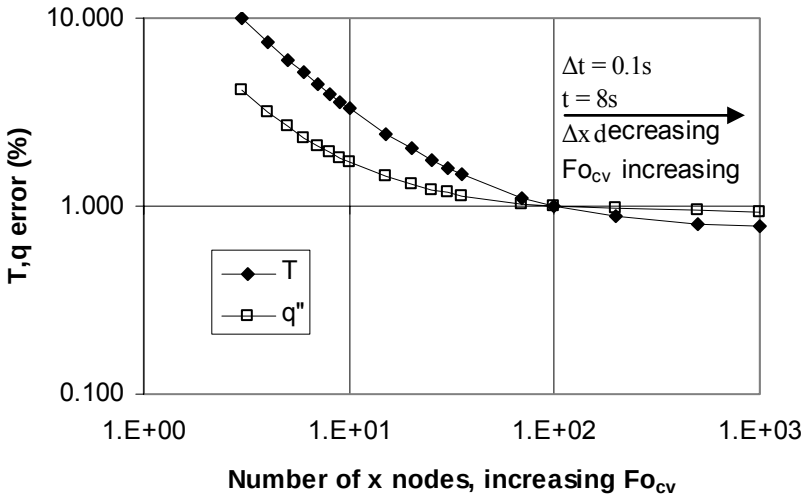
Figure (6a) shows the inverse-predicted temperature and heat flux error at t = 8 s holding the x-direction spacing constant with 100 nodes ($\Delta x = 0.0503$ mm) and a varying number of time steps. This figure shows an increase in accuracy as both the time step size and the nodal Fourier number, $Fo_{cv} = \alpha\Delta t/\Delta x^2$ are decreased. Figure (6b) shows the inverse-predicted temperature and heat flux error at t = 8 s holding the time spacing constant with 100 time steps ($\Delta t = 0.1$ seconds) and a varying x-direction spacing. This figure demonstrates the accuracy of the solution is improved by decreasing the nodal spacing, which increases the nodal Fourier number.

Figure (6) suggests that refining both the spatial and temporal meshes improves accuracy, but this shows a contradiction for the effect of the nodal Fourier number: the accuracy was improved by both increasing and decreasing the Fourier number. This contradiction is again seen in Figure (7a). For a given nodal Fourier number, both good and poor accuracy can be obtained, depending on the choice of nodal spacing and time step size. Likewise, good (or poor) accuracy can be obtained with a low or high nodal Fourier number. Therefore, the nodal Fourier number has no effect on accuracy.

Realizing that the Fourier number plays an important role in heat conduction problems, the role of the Fourier number was further investigated. The time step Fourier

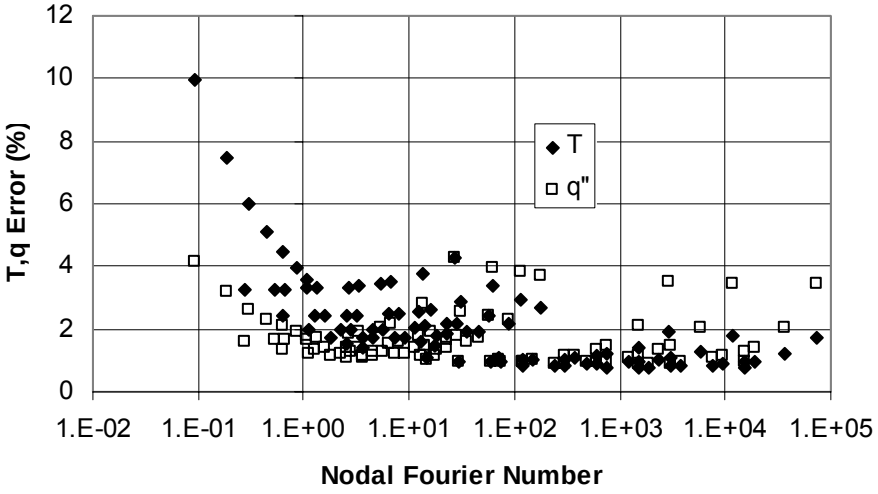


(a)

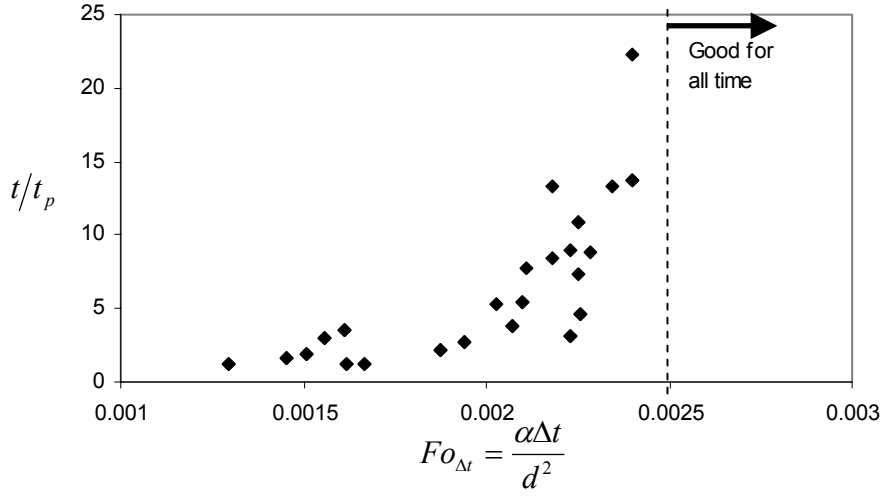


(b)

Figure 6: Dependence of accuracy on (a) number of time steps and (b) number of spatial nodes for the one-dimensional inverse-predicted surface temperature and heat flux error using perfect data (ε = 0%) from the direct problem.



(a)



(b)

Figure 7: Effect of Fourier number on the inverse solution: (a) effect of nodal Fourier number, $Fo_{cv} = \alpha\Delta t/\Delta x^2$, on accuracy and (b) effect of $Fo_{\Delta t} = \alpha\Delta t/d^2$, on stability. t_p was 0.88s.

number was defined as the $Fo_{\Delta t} = \alpha \Delta t / d^2$, where d is the distance from the sensor to the surface. The inverse code was run for varying sensor depths, time step sizes and nodal spacing. For some combinations of these parameters, the inverse prediction was found to be unstable at the surface. The nodal spacing was found to play no role in the stability of the solution. This can be seen in Figure (7b), which is a plot of dimensionless time versus $Fo_{\Delta t}$. This plot shows that for $Fo_{\Delta t}$ less than 0.0025, the inverse code produced a stable solution for a finite amount of time before the solution blew up. At the instant the solution at the surface became unstable, the prediction one node below the surface was still stable. Figure (7b) shows how far in time and space the inverse code can project with stability. Physically, this is a plot of how far in time the inverse code will produce a stable prediction for a given time step size and sensor depth. A choice of $Fo_{\Delta t} > 0.0025$ produced a stable prediction for all time and for any sensor depth, and so is the stability criterion for the 1D inverse method.

This is not to say that the prediction achieved with a high time step Fourier number will be accurate. As shown in Figures (6-7), the spatial and temporal mesh should be refined to achieve accuracy. Therefore, both accuracy and stability were achieved by refining the spatial mesh, and by refining the temporal mesh until the $Fo_{\Delta t}$ was just above 0.0025.

4.2 Noisy, Filtered Data.

The results were again found to vary with the chosen time and space mesh. Applying the knowledge learned from the case of perfect data, spacing in the x-direction,

as well as the time step size were investigated to see if the trends observed when using perfect data still applied.

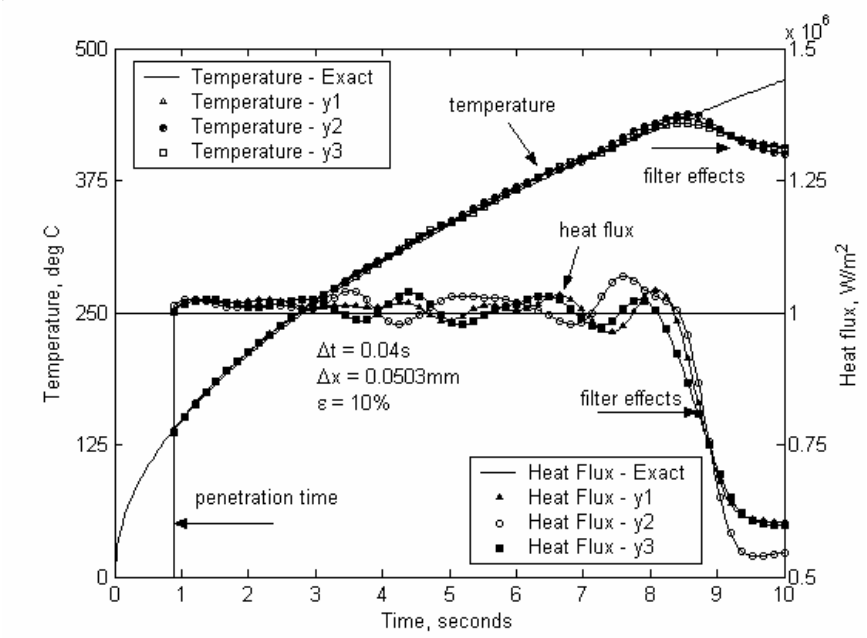
Noise with a uniform distribution (white noise) was added to the "perfect" data using the profile given by

$$\widetilde{T}(y_j,t_k)=\left[1+u_{j,k}\varepsilon\right]T_s(y_j,t_k),\quad j=-N,-N+1,...,N-1,N\quad k=0,1,...,P \qquad (12a)$$

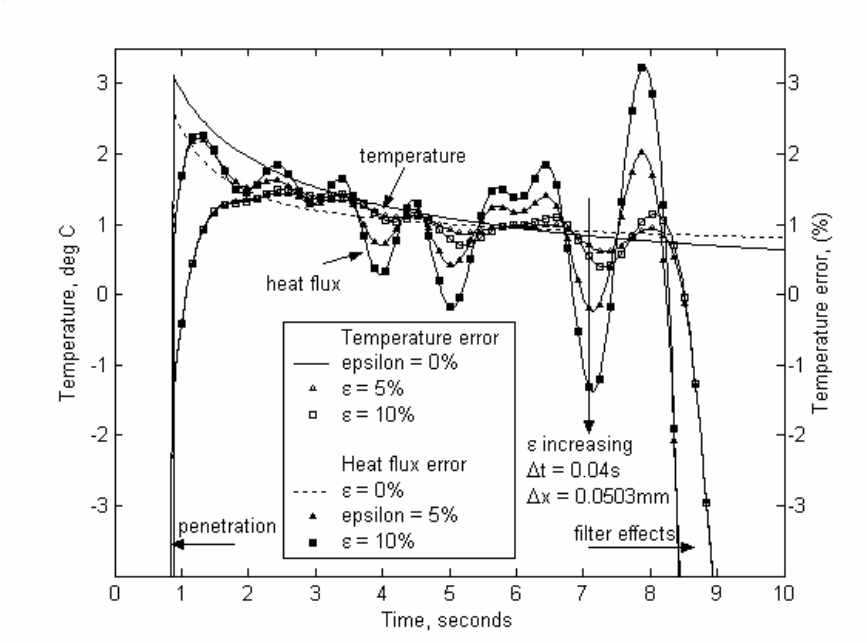
$$\widetilde{q}_x^{"}(y_j,t_k)=\left[1+u_{j,k}\varepsilon\right]q_{x,s}^{"}(y_j,t_k),\quad j=-N,-N+1,...,N-1,N\quad k=0,1,...,P \qquad (12b)$$

where $\widetilde{T}$ and $\widetilde{q}_x^{"}$ are the noisy sensor data, T_s and $q_{x,s}^{"}$ are the perfect sensor data, ε is the noise factor in percent of signal, $u_{j,k}$ is a random number between -1 and 1 based on uniform distribution, P is the number of time steps, and 2N+1 nodes in the y-direction are used. The resultant data were filtered using a Gauss filter described above with a cutoff frequency of 0.5 Hz to filter out unwanted noise. For a discussion on the filtering technique used in this thesis, including choice of cutoff frequency, see Appendix A. The data was padded with two seconds of initial data to improve the filter output. The ending data (t > 8.3s) was found to be affected by the filter and diverged from the correct solution. This was a result of Gauss filter and so was removed from the final solution data. This noise was added to each sensor individually, so that each sensor received a different noise distribution. Therefore, all three nodes in the y-direction at a given x-location no longer possess identical information. The inverse solution was plotted in Figure (8a) where differences across nodes y1, y2, and y3 can be seen. Because of this effect, the average surface temperature and heat flux are presented below.

The effect of input error, ε, was also of interest. Values of input error, ε, were varied to 0%, 5%, and 10%. Figure (8b) shows the effect of the input error on the



(a)



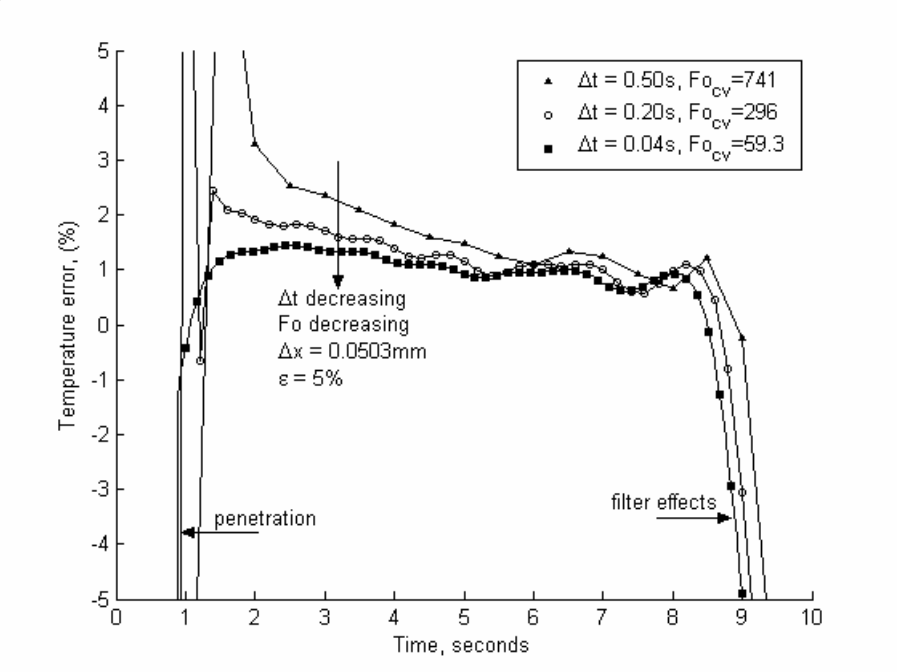
(b)

Figure 8: One-dimensional inverse-predicted surface temperature and heat flux histories using noisy, filtered data. (a) Differences across the three surface nodes, y1, y2 and y3. (b) Effect of input error, ε on the temperature and heat flux error histories calculated using the average of y1, y2, and y3.

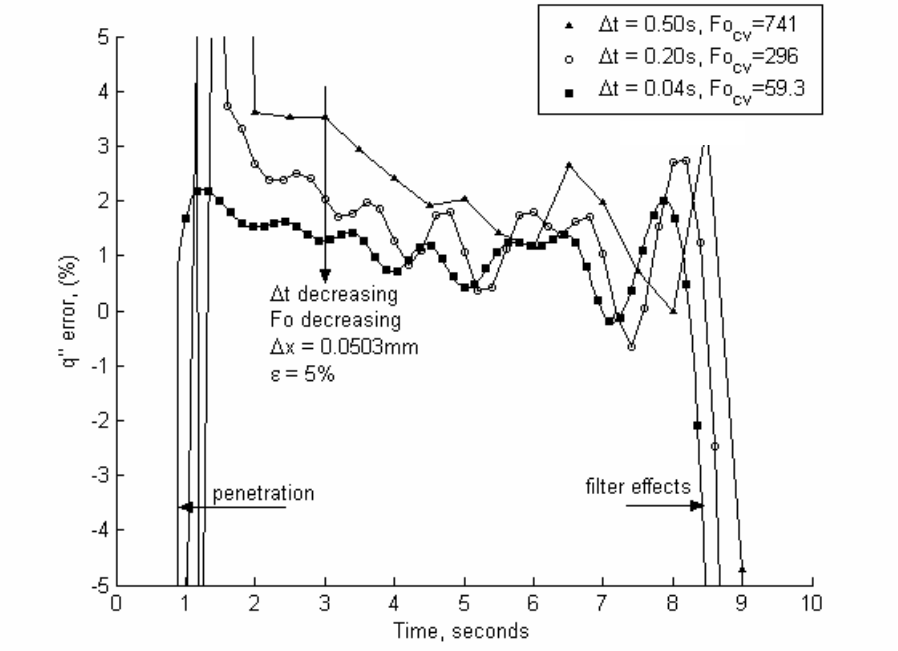
temperature and heat flux solutions. The heat flux solution was found to be accurate to 3.5% for an input error of up to 10% for $t < 8s$. This amazing result effectively damped out the measurement error, meaning that the 1D problem behaved as if it was well-posed.

Figure (9) shows the effect of time step size on the inverse-predicted temperature and heat flux error. These two plots show that while the solutions are oscillatory in time, refining the temporal mesh produced better results at any given time step. Similarly, Figure (10) shows the effect of the spatial mesh on the inverse solution. Again, while the solutions were oscillatory in time, refining the spatial mesh produced better results at any given time step. The effect of the nodal Fourier number on accuracy is again conflicting: decreasing Fo_{cv} in Figure (9) improved accuracy, while increasing Fo_{cv} in Figure (10) improved accuracy. As with the perfect data, the nodal Fourier number had no effect on the accuracy of the solution. These conclusions are reaffirmed in Figures (11-12). It is seen in Figure (11) that for a given input error, the accuracy of the surface temperature and heat flux solution improved as the number of time steps used was increased. Similarly, Figure (12) shows the inverse solution improved as the spatial mesh was refined. Figures (11-12) again show the ineffectiveness of the nodal Fourier to predict accuracy.

It is important to note here the difference between stability and accuracy. Figure (7b) suggests that a larger Δt should be used for stability, while Figures (9,11) suggest a finer Δt should be used for accuracy. More insight on this issue can be drawn from a more thorough examination into the effect of a coarse Δt seen in Figure (9). Note that for a fine $\Delta t = 0.04s$, the inverse method is able to predict an accurate solution at the penetration time, $t_p = 0.88s$. However, the coarser $\Delta t = 0.5s$ is unable to predict an

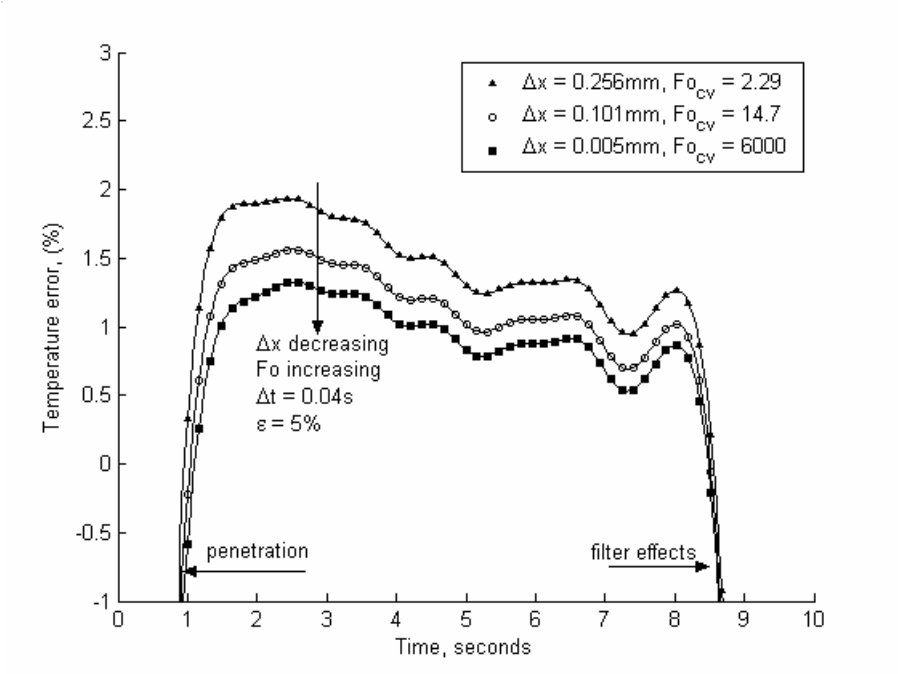


(a)

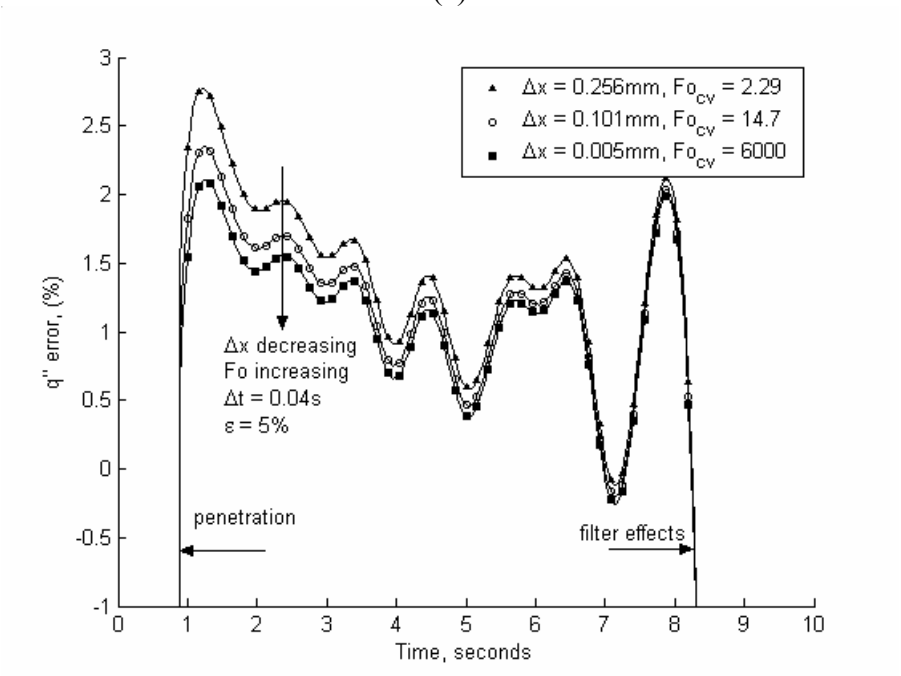


(b)

Figure 9: Effect of Δt on the one-dimensional inverse-predicted surface (a) temperature error and (b) heat flux error histories using noisy (ε = 5%), filtered data.

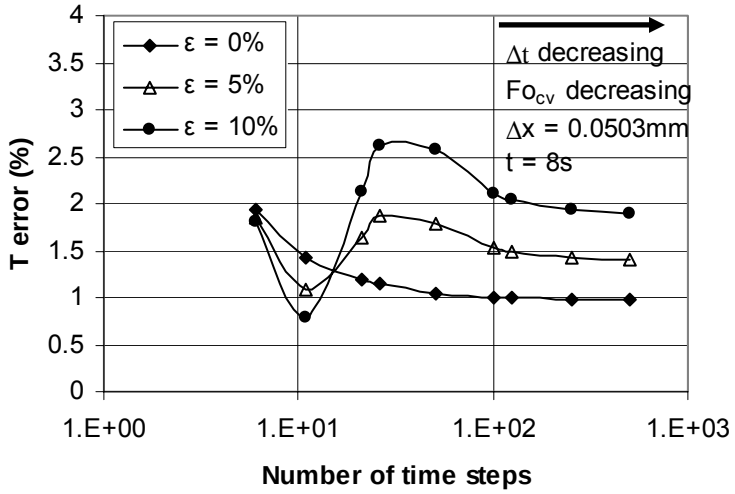


(a)

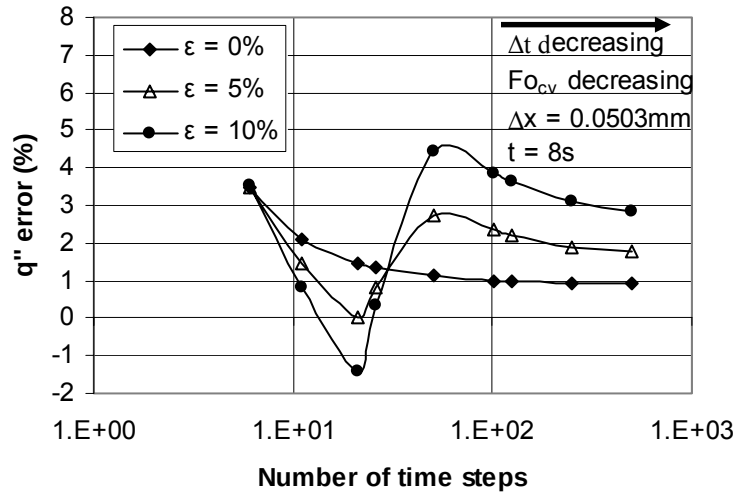


(b)

Figure 10: Effect of Δx on the one-dimensional inverse-predicted surface (a) temperature error and (b) heat flux error histories using noisy (ε = 5%), filtered data.

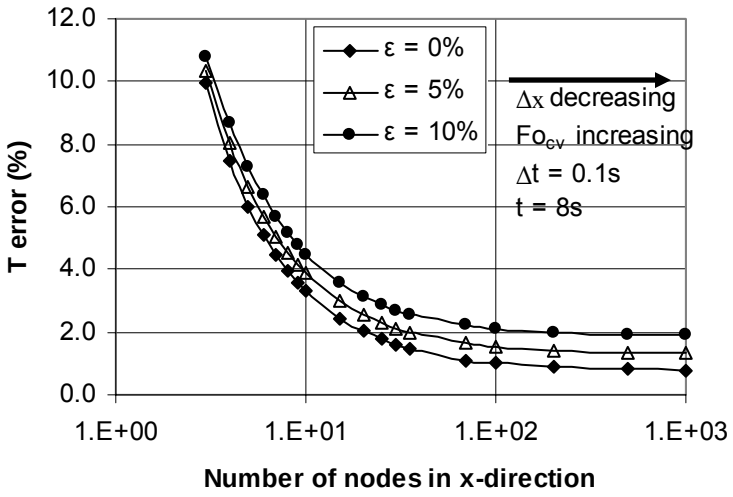


(a)

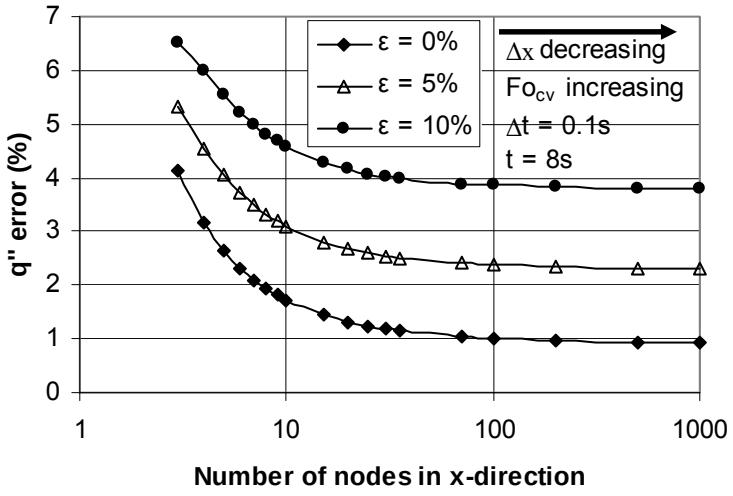


(b)

Figure 11: Effect of time step size of the one-dimensional inverse-predicted surface (a) temperature error and (b) heat flux error at t = 8 seconds using noisy, filtered data.



(a)



(b)

Figure 12: Effect of Δx on the one-dimensional inverse-predicted surface (a) temperature error and (b) heat flux error at t = 8 seconds using noisy, filtered data.

accurate solution until t = 2.0s, more than twice the penetration time. Therefore, a balance between a coarse Δt for stability and a fine Δt for accuracy should be used. In fact, the most accurate, yet stable choice of Δt would be such that the $Fo_{\Delta t}$ is just slightly larger than 0.0025.

It can be concluded the one-dimensional, noisy data followed the same trend as the one-dimensional perfect data. The optimum solution from this method can be found by refining both the spatial and temporal meshes within stability requirements until convergence is found. This is a striking difference from most inverse methods which require a large Δt to maintain stability and accuracy.

4.3 Use of heat flux – heating rate relationship

The heat flux – heating rate relationship given in Equation (3) was then applied to the temperature data to obtain the heat flux data. This is the final step in proving the inverse method for the one-dimensional problem. Figure (13) shows the result of using only temperature data from the direct code and Equation (3) to obtain the sensor heat flux. An input error of 10% was used and filtered. Good results were seen with a surface temperature error less than 1% for $2 < t < 8$s and a surface heat flux error of less than 5% for $2 < t < 7.5$s. Again, the inverse code reduced the measurement error in half, behaving like a well-posed problem. This is amazing accuracy for the ill-posed inverse problem and showed good promise for the two-dimensional method.

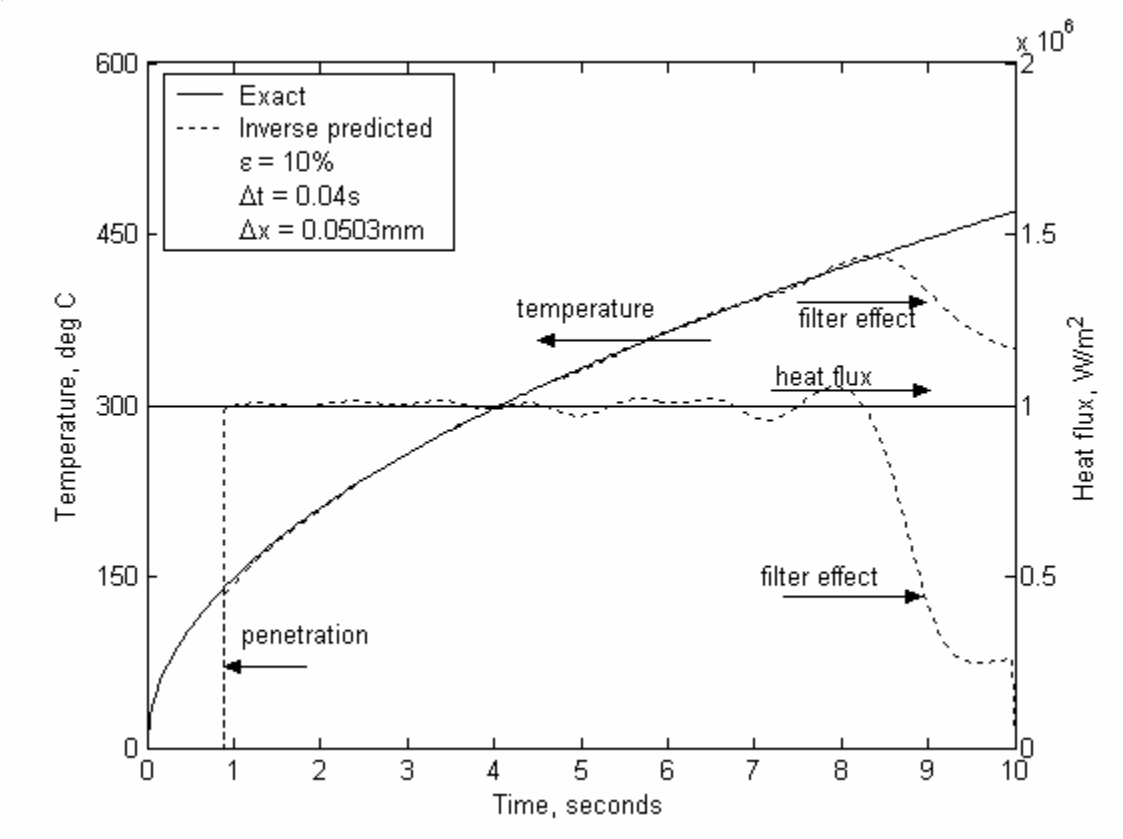


Figure 13: One-dimensional inverse solution using Equation (3) to obtain the sensor heat flux with 10% input error. Only temperature data from the direct code was used to obtain the inverse solution.

Chapter 5 Two-Dimensional Results

A two-dimensional boundary condition was imposed at the north face seen in Figure (2a). The heat flux on the surface is given by

$$q_x''(0,y,t) = F(y) = \begin{cases} q_0'', & |y| \le b \\ 0, & |y| > b \end{cases}, \quad t > 0 \tag{11}$$

This condition was chosen so as to utilize the exact solution [3] for the surface temperature given by

$$T(0,y,t) = \frac{q_0''\sqrt{\alpha}}{\sqrt{\pi}k}\left\{\sqrt{t}\left[erf\left(\frac{y+b}{2\sqrt{\alpha t}}\right)+erf\left(\frac{b-y}{2\sqrt{\alpha t}}\right)\right]+\frac{y+b}{2\sqrt{\alpha\pi}}\Gamma\left(0,\frac{(y+b)^2}{4\alpha t}\right)+\right.$$

$$\left.\frac{b-y}{2\sqrt{\alpha t}}\Gamma\left(0,\frac{(b-y)^2}{4\alpha t}\right)\right\}, \quad y\in(-\infty,\infty), \quad t \ge 0 \tag{12}$$

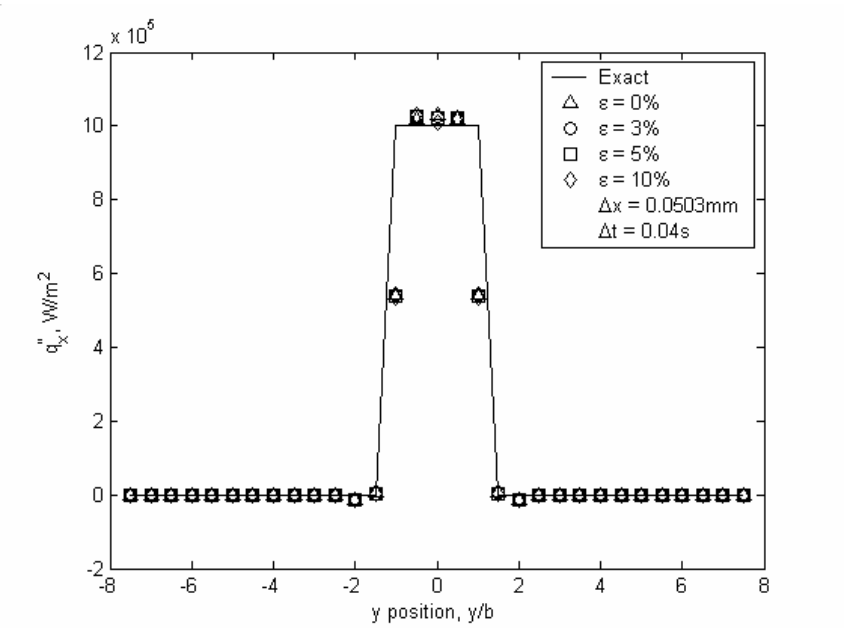
where erf(z) is the error function and Γ(0,z) is the incomplete Gamma function [4]. The depth of the sensor, d, was taken to be 0.005 m and the half-width of the heat source, b, was taken to be 0.01 m. Temperatures were calculated over the domain for 10 seconds. Temperature values were found to be sufficiently close to zero (10^{-7} ^{o}C as compared to peak value of 440 ^{o}C) over a radius of 6b from the origin at t_{max}. Therefore, the computational domain seen in Figure (2a) was used for the forward problem.

With the end goal of using the heat flux – heating rate relationship to determine the heat flux, the temperature and heat flux data at x=d was first extracted from the forward problem and used to recreate the surface temperature and heat flux using the inverse method described above. For the current example, the explicit direct method was

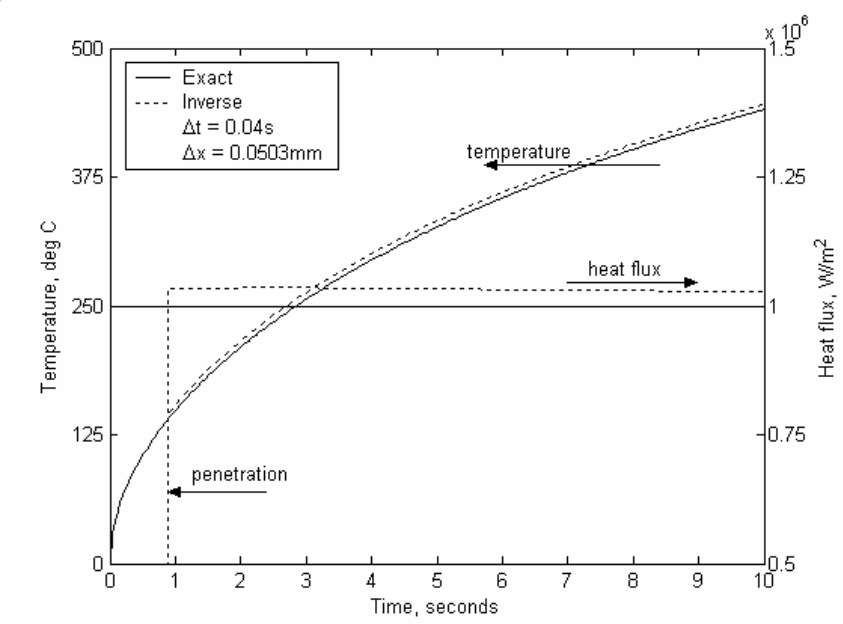
used with 151 spatial nodes in the x-direction, 301 spatial nodes in the y-direction, and 10001 temporal nodes. This mesh was chosen so as to minimize the surface temperature error, which was kept below 1% for t > 1.5 seconds. The maximum spacing of 0.5 cm previously specified was used as the distance between sensors. Data at these points were exported for use in the inverse problem, and all other data discarded. The domain used for the inverse problem can be seen in Figure (4b).

5.1 Two-Dimensional Perfect Data

In order to further validate the inverse code, the raw sensor data with no input error ("perfect" data) was used as input to the inverse code. Figure (14a) shows the resultant surface heat flux distribution and error at t = 8s. Note Figure (14a) includes solution data using measurement error which will be discussed in Section 5.2. One can see the accuracy of the inverse method in reproducing the surface heat flux; the error observed for center nodes y1, y2, and y3 was less than 3%. Since y1, y2 and y3 are attempting to reproduce the magnitude of the pulse, the average of the three nodes, q''_{av}, was used for heat flux error determination. The temperature history at the center node and the average heat flux history can be seen in Figure (14b), which shows an accurate prediction of the heat flux and temperature. The same stability requirement of $Fo_{\Delta t} > 0.0025$ was also applicable to the two-dimensional case. After investigating the sensor data, the dimensionless temperature response, θ, was again found to be 0.01 at t = 0.88 seconds; therefore, this value was used as the penetration time, t_p, for the 2D problem.



(a)



(b)

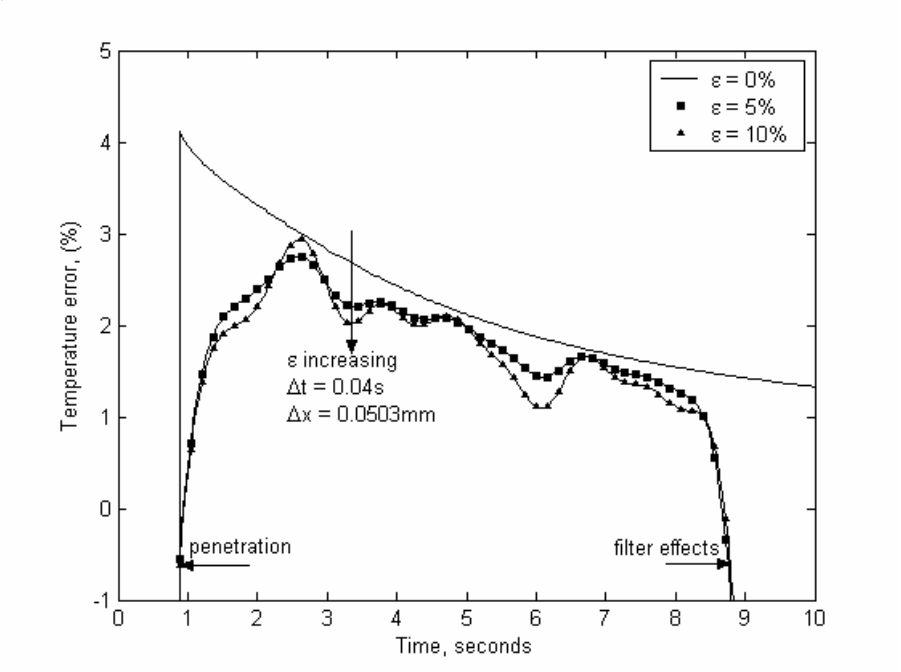
Figure 14: Two-dimensional surface heat flux and temperature solutions. (a) The accurate reproduction of the heat flux distribution across the north boundary at t = 8s using up to 10% input error. (b) The resultant surface temperature history at y = 0 and average heat flux histories of nodes y1 = -Δy, y2 = 0, and y3 = Δy using perfect (ε = 0%) data.

5.2 Two-Dimensional Noisy Data

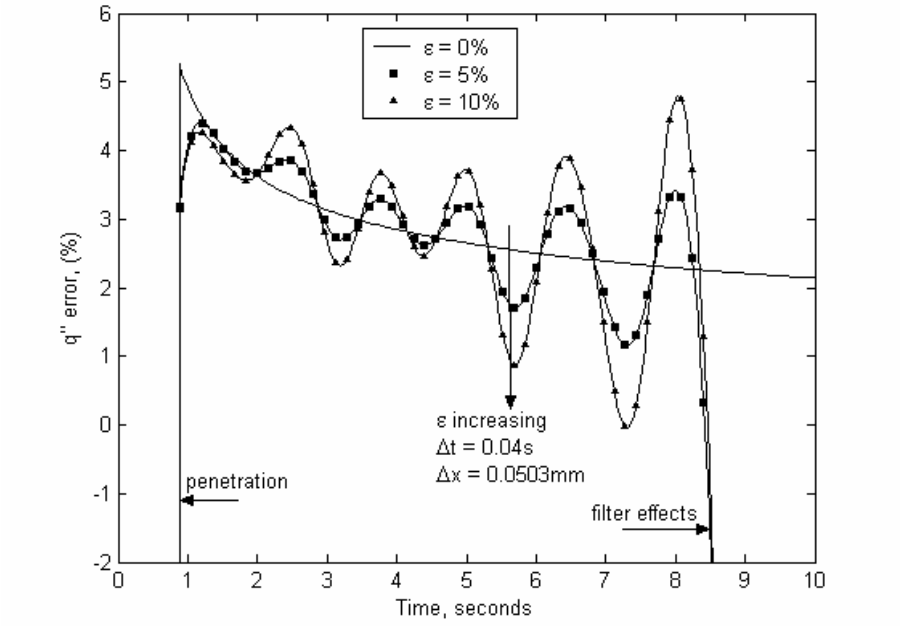
Following the same analysis as the one-dimensional noisy case, the effects of adding noise to the two-dimensional problem was investigated. Again, a Gauss low pass filter was used with a cutoff frequency of 0.5 Hz. For a discussion on filtering technique, including choice of cutoff frequency, see Appendix A. Figure (14a) shows the resultant surface heat flux distribution at t = 8s for varying degrees of input error, ε, with Δx and Δt fixed. Note that for an input error of up to $\varepsilon = 10\%$, a favorable surface heat flux distribution is obtained. In fact, $\varepsilon = 10\%$ looks identical to the $\varepsilon = 0\%$ case on this plot.

Figure (15) shows the resultant temperature and heat flux error histories for varying degrees of input error, ε, holding Δx and Δt fixed. Clearly, the $\varepsilon = 0\%$ case (perfect, unfiltered data) shows the bias of the 2D inverse method. The $\varepsilon = 5,10\%$ heat flux error histories oscillate about the $\varepsilon = 0\%$ case. Figure (16) shows the temperature and heat flux error histories with the $\varepsilon = 0\%$ solution error subtracted. This can be read as the actual effect of measurement error to the data. As in the 1D case, the inverse code reduced the magnitude of the measurement error: an input error of $\varepsilon = 10\%$ yielded a temperature solution (bias removed) bounded by 1% and a heat flux solution (bias removed) bounded by 3% for for $2s \leq t \leq 8s$.

Seeking to verify the stability and accuracy of the spatial and temporal mesh, analysis similar to the one-dimensional case was conducted. First investigating the effects of the temporal mesh, the temperature error and heat flux error histories were plotted in Figure (17). Again, one sees that increasing the number of temporal nodes increased accuracy for both temperature and heat flux.

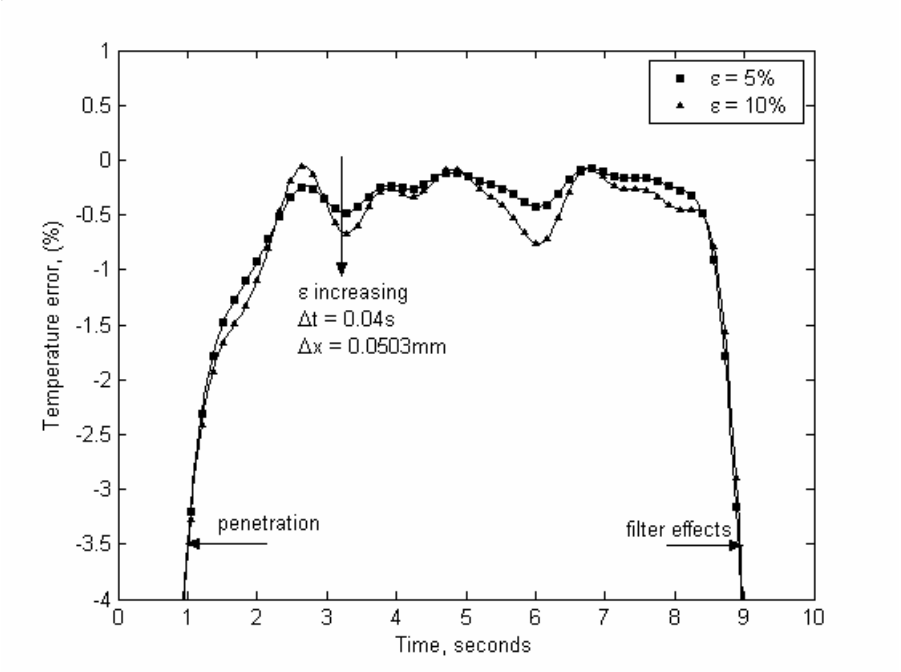


(a)

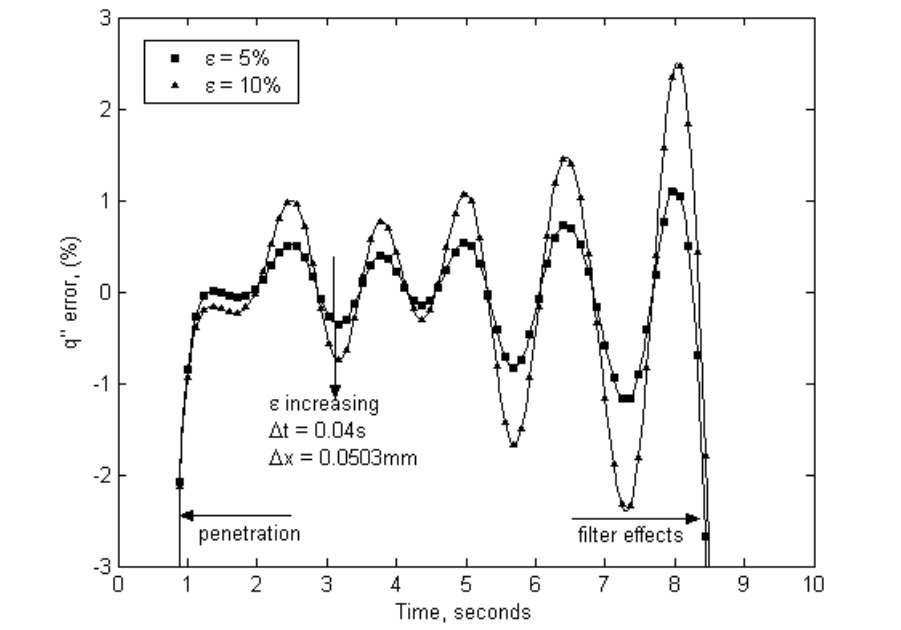


(b)

Figure 15: Effect of input error, ε, on the two-dimensional inverse-predicted surface (a) temperature error at y = 0 and (b) average heat flux error histories of nodes y1 = -Δy, y2 = 0, and y3 = Δy using noisy, filtered data.

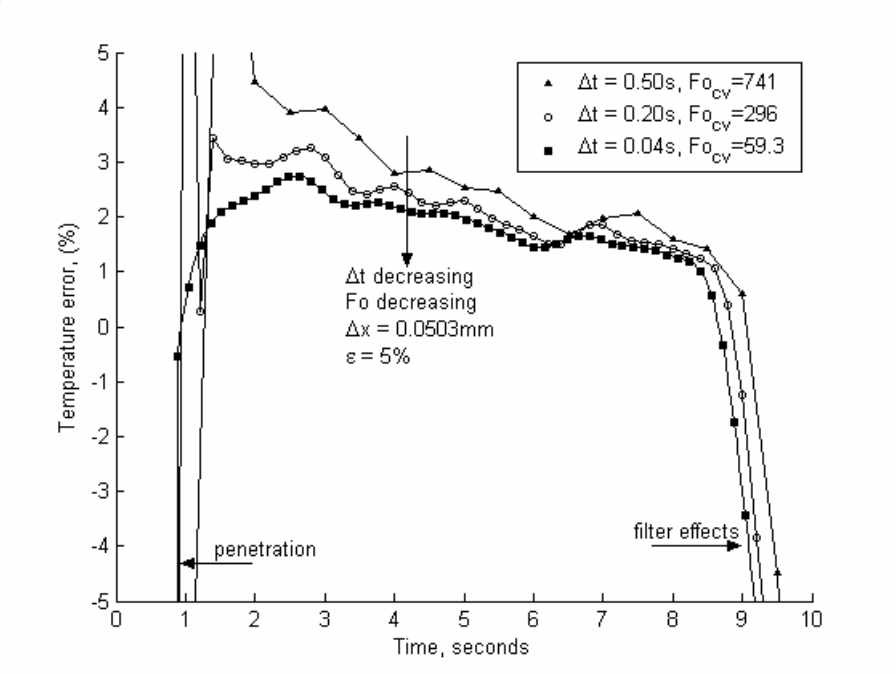


(a)

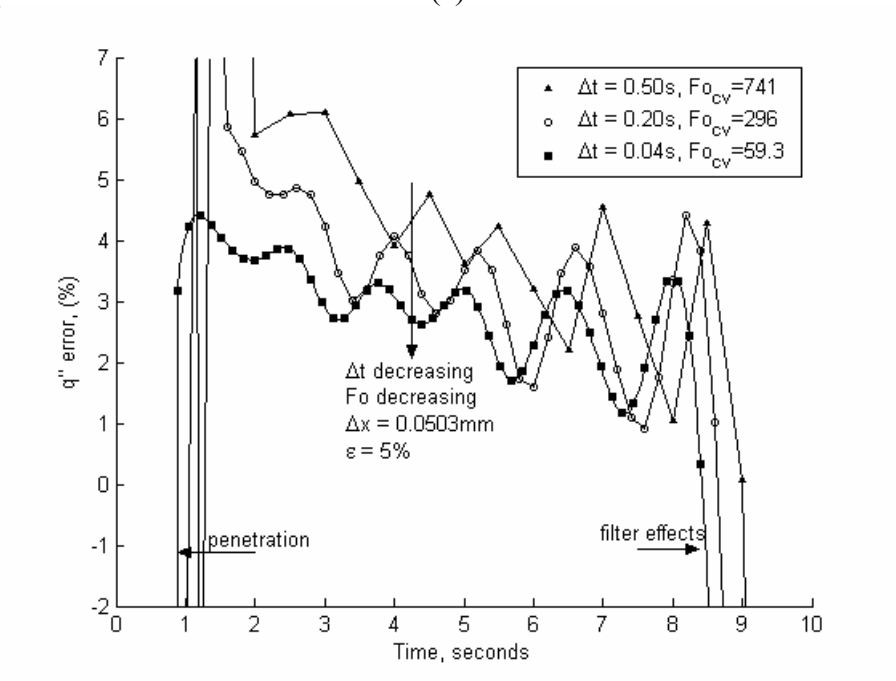


(b)

Figure 16: Effect of input error, ε, with bias removed for 2D inverse problem. (a) Temperature history and (b) heat flux history.



(a)



(b)

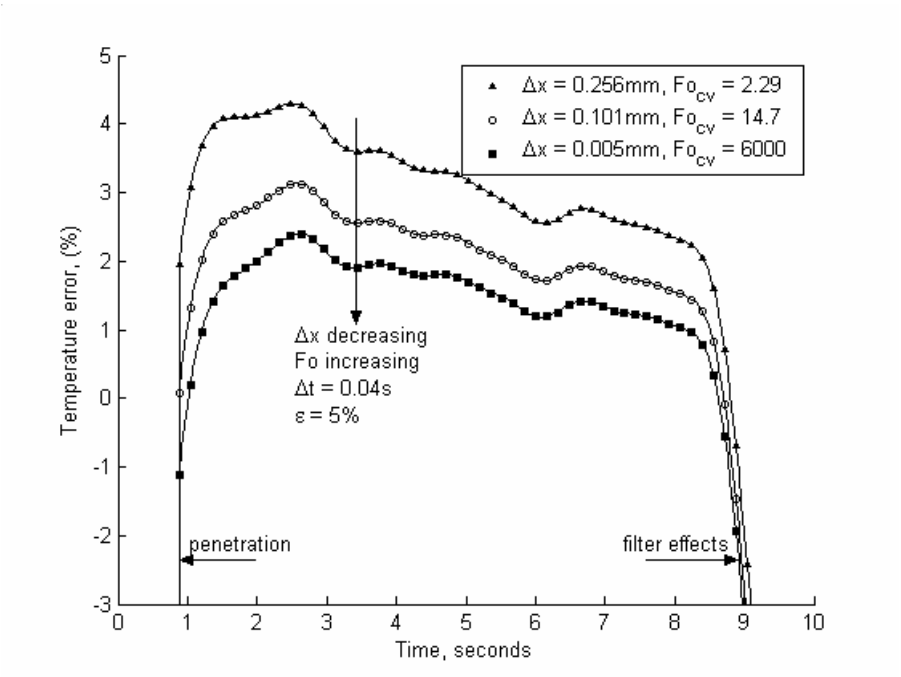
Figure 17: Effect of Δt on the two-dimensional inverse-predicted surface (a) temperature error at y = 0 and (b) average heat flux error histories of nodes y1 = -Δy, y2 = 0, and y3 = Δy using noisy (ε = 5%), filtered data.

Looking back to the stability criterion seen in Figure (7b), a larger time step size achieves stability. However, Figure (17) shows the time step size should be decreased for accuracy. A time step size of 0.04s predicted an accurate inverse solution for $t > t_p$. However, as Δt became coarser, the accuracy of the inverse solution improves at a later time. The time step size of 0.50s required two seconds, more than twice the penetration time, before it was able to make an accurate inverse prediction. Therefore, a balance must made, and the time step size should be decreased until $Fo_{\Delta t}$ is slightly greater than 0.0025 in order to achieve both accuracy and stability.

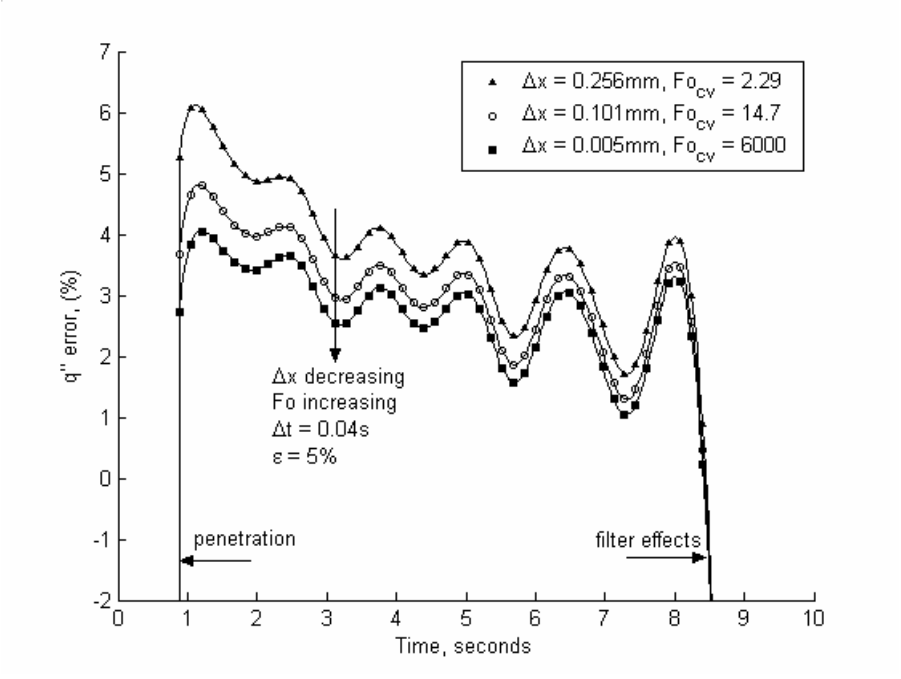
The effects of the spatial mesh on the temperature error and heat flux error histories were plotted in Figure (18). Similar to the one-dimensional case, increasing the number of spatial nodes increased accuracy. This provided the same contradiction for the nodal Fourier number as seen in the one-dimensional case; the accuracy was improved by decreasing Fo_{cv} in Figure (17) and by increasing Fo_{cv} in Figure (18). Therefore, the accuracy of the 2D inverse solution was independent of the nodal Fourier number, but depends heavily on the spatial and temporal mesh.

5.3 Use of heat flux – heating rate relationship

The final piece of the puzzle was to use Equation (5) to obtain the sensor heat flux provided only temperature data from the direct problem. An input error of 5% was used for the temperature data and filtered using the Gauss filter. The use of Equation (5) required more points in the y-direction than the restriction of $\Delta y = 0.5$cm allowed. Therefore, the sensor data was also filtered in the y-direction with a cutoff frequency of



(a)



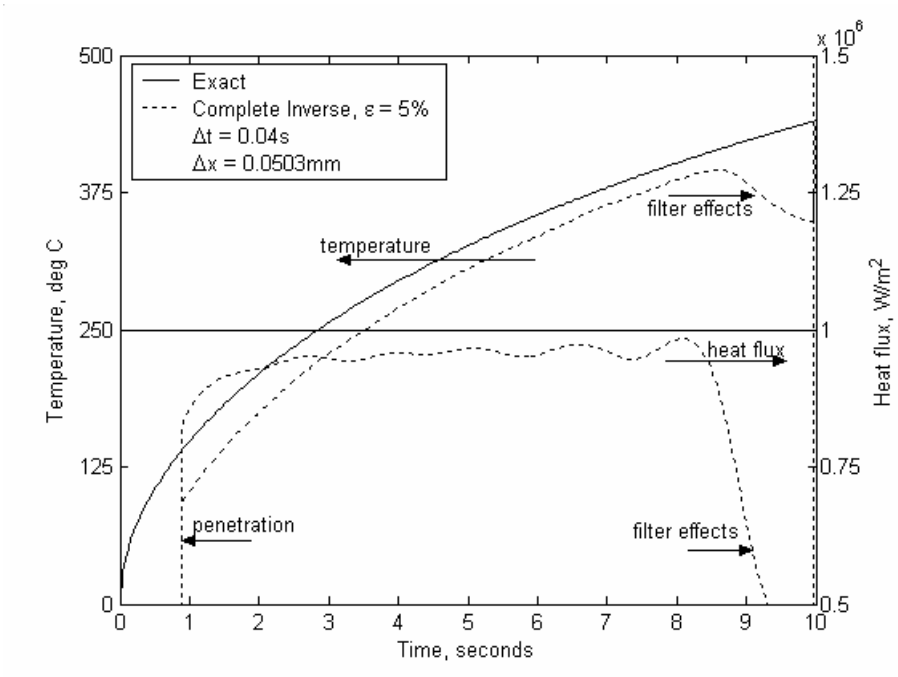
(b)

Figure 18: Effect of Δx on the two-dimensional inverse-predicted surface (a) temperature error at y = 0 and (b) average heat flux error histories of nodes y1 = -Δy, y2 = 0, and y3 = Δy using noisy (ε = 5%), filtered data.

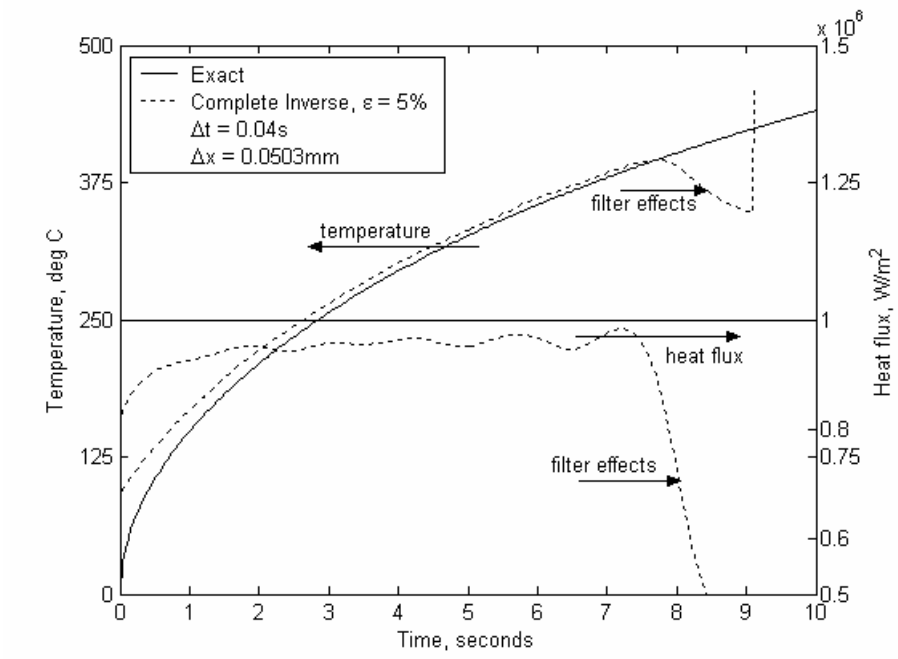
100 Hz to allow the number of nodes in the y-direction to be increased. A total of 301 y-nodes were used to calculate the heat flux using Equation (5). This number was reduced to 31 nodes for the inverse problem to be comparable to the previous analysis.

Figure (19) shows the resultant surface temperature and flux histories for an input error of 5%, resulting in a heat flux error of only 5%. It is interesting to note that if the inverse solution is shifted backwards in time by the penetration time of 0.88 seconds as seen in Figure (19b), the temperature history error improves dramatically. Note the results from Figure (19) used the same temperature data as the results in Figures (14-18). The difference is the sensor heat flux used. A comparison of the different heat fluxes used is shown in Figure (20). Note the integral obtained heat flux appears to be shifted forward in time by approximately t_p. The reason for this is as of yet undetermined, and, as discussed previously, the effect of the penetration time is an issue that requires further investigation, including alternate integration schemes. Figure (21) shows the resultant surface heat flux distribution was predicted accurately across all nodes.

It is worth reiterating that this solution was obtained using only one line of temperature data and a forward difference to obtain the heating rate. This data were then used to obtain the sensor heat flux and project to the surface. While these results were not quite as accurate as using the heat flux data directly from the forward problem, the process of instrumenting one line of thermocouples is significantly simpler than using thermocouples and heat flux gauges. This approach only slightly affected the accuracy of the results, and the inverse method proved to be well-behaved; the magnitude of the measurement error was not increased by the inverse method.



(a)



(b)

Figure 19: Inverse solution using the heat flux – heating rate relationship. Input error of 5% used to calculate solution.

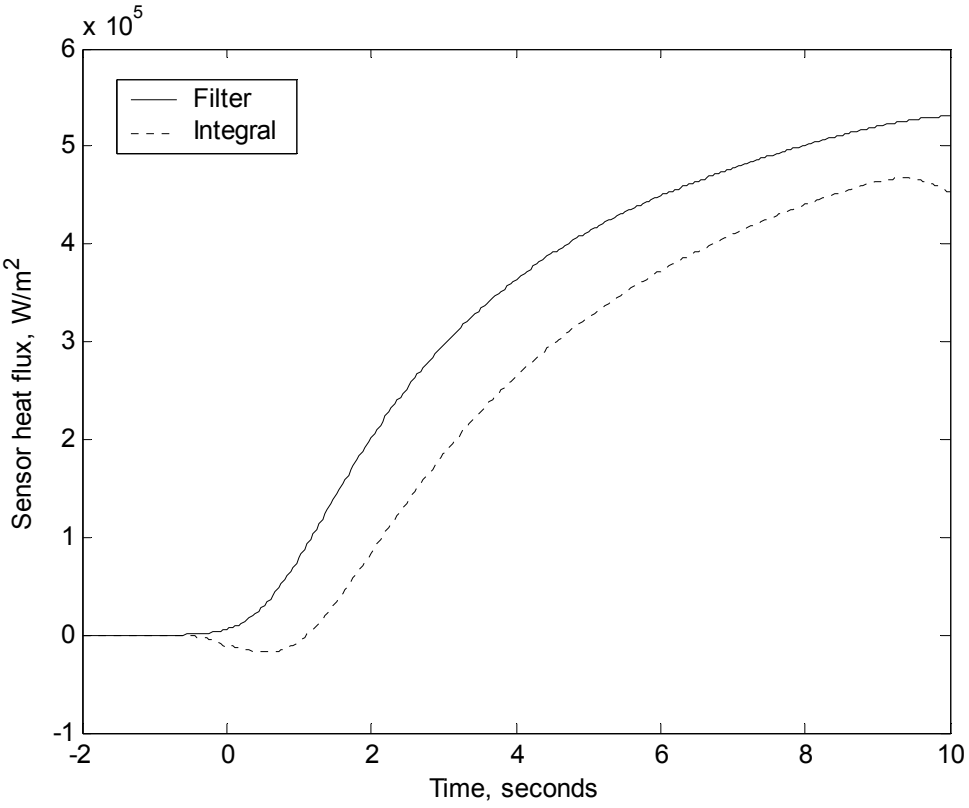


Figure 20: Comparison of sensor heat flux from direct problem and obtained using Equation (5).

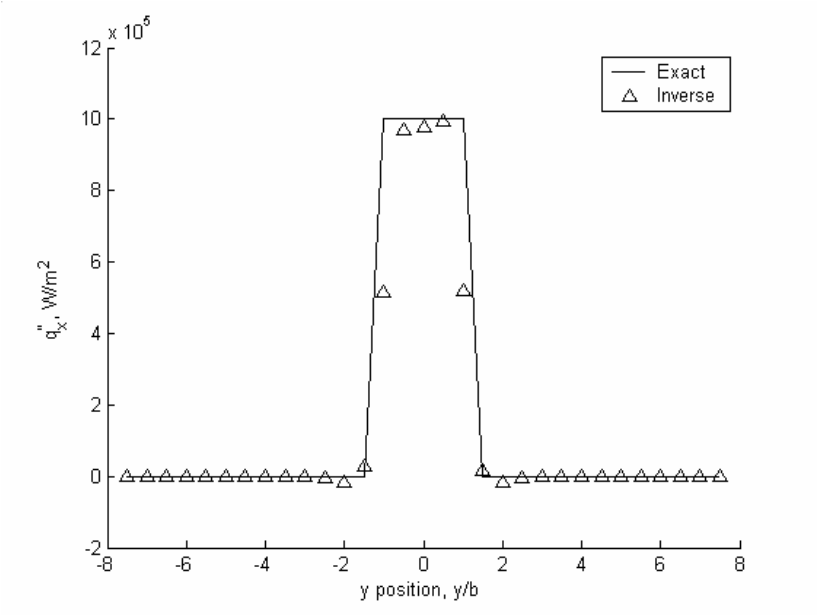


Figure 21: Inverse surface heat flux distribution using the heat flux – heating rate relationship using $\varepsilon = 5\%$.

Chapter 6 Conclusions

A new method of solving the inverse heat conduction problem has been presented and proven accurate and stable. A simple implicit in time with space marching approach was used in combination with digital filtering for regularization. One-dimensional perfect data results showed decreasing the time step size, which decreased the nodal Fourier number, provided more accurate results. The data also showed refining the spatial mesh, which increased Fo_{cv}, improved the accuracy. This conflicting result for the effect of Fo_{cv} showed that the Fourier number had no effect on accuracy. It was found, however, the Fourier number played a role in stability. The time step Fourier number, $Fo_{\Delta t}$, greater than 0.0025 proved to be unconditionally stable. Therefore, for accuracy and stability the temporal mesh should be refined until this limit on $Fo_{\Delta t}$ is reached.

One-dimensional results with noisy input data demonstrated the effectiveness of the digital filtering technique. The filter employed a cutoff frequency as the regularization parameter, which has an exact definition with physical significance. Measurement error of 10% was added to the data, filtered, and an inverse solution with only 3.5% heat flux error was predicted. The prediction error resulting from the use of "perfect" data suggested a bias (over-prediction). If the error bias was removed from the prediction, the heat flux error was reduced to 2%. The dramatic reduction of error through an inverse process was striking, since the inverse problem is well-known to magnify measurement errors.

Two-dimensional results echoed the results from the one-dimensional case. Using temperature and heat flux data from the direct results with 10% measurement error, the inverse method produced a heat flux error of only 3% (bias removed). An additional case using only one line of temperature data and no heat flux data from the direct problem was used as alternate approach to the inverse problem. An integral relating heat flux and the heating rate on the half-space was employed using only this temperature data to obtain the heat fluxes at the sensor sites. Results from this method with a measurement error of 5% predicted a surface heat flux error of only 5%. While not as accurate as the case where heat flux was directly exported from the direct problem, this approach eliminates the need for heat flux gauges at the sensor site. Results utilizing this integral relationship required a shift backwards in time by the penetration time, the reason for which requires further investigation. The two-dimensional results proved the accuracy and stability of the new method. The present method could easily be applied to three dimensions.

List of References

List of References

[1] Beck, J.V., Blackwell, B., St. Clair, C.R., Jr. *Inverse Heat Conduction*. Wiley, New York, 1985.

[2] Frankel, J.I., "Regularization of Inverse Heat Conduction by Combination of Rate Sensor Analysis and Analytic Continuation," *Journal of Engineering Mathematics*, 57, 2007, pp. 181-198.

[3] Incropera, F.P. and DeWitt, D.P., *Fundamentals of Heat and Mass Transfer*, John Wiley & Sons, Inc. New York. 5th edition.

[4] Abramowitz, M. and Stegun, I.A., *Handbook of Mathematical Functions*, Dover, New York, 1965.

[5] Frankel, J.I., Keyhani, M., Arimilli, R.V., and Wu, J., "A New Multidimensional Integral Relationship between Heat Flux and Temperature for Direct Internal Assessment of Heat Flux", *Zeitschrift für Angewandte Mathematik und Physik, ZAMP*, 2007; 58:1-20.

[6] Carasso, A.S., "Slowly Divergent Space Marching Schemes in the Inverse Heat Conduction Problem," *Numerical Heat Transfer B*, Vol. 23, 1993, pp.111–126.

[7] Carasso, A.S., "Space Marching Difference Schemes in the Nonlinear Inverse Heat Conduction Problem," *Inverse Problems*, Vol. 8, 1992, pp. 25–43.

[8] Hadamard, J., *Lectures on Cauchy's Problem in Linear Partial Differential Equations*, Yale University Press, New Haven, CT, 1923.

[9] Burggraf, O.R., "An Exact Solution of the Inverse Problem in Heat Conduction Theory and Applications," *ASME J. Heat Transfer*, 86C, pp. 373-382, August 1964.

[10] Tikhonov, A.N. & Arsenin, V.Y., *Solutions of Ill-Posed Problems*, V.H. Winston & Sons, Washington, D.C., 1977.

[11] Chen, H.T., Lin, S.Y., and Fang, L.C., "Estimation of Surface Temperature in Two-dimensional Inverse Heat Conduction Problems," *International Journal of Heat and Mass Transfer,* Vol. 44, No. 8, 2001, pp. 1455.

[12] Taler, J. and Zima, W., "Solution of Inverse Heat Conduction Problems Using Control Volume Approach," *International Journal of Heat and Mass Transfer*, Vol. 42, 1999, pp. 1123–1140.

[13] Kulish, V.V., & Novozhilov, V.B., "Integral Equation for the Heat Transfer with the Moving Boundary," *AIAA Journal of Thermophysics and Heat Transfer*, 17:538-540.

[14] Kulish, V.V. and Lage, J.L., "Fractional Diffusion Solutions for Transient Local Temperature and Heat Flux," *Heat Transfer* 122:372-376.

[15] Patankar, S.V., *Numerical Heat Transfer and Fluid Flow*, Hemisphere Publishing Corp., Washington, DC, 1980.

[16] Grigull, U. and Sandner, H., *Heat Conduction*, Springer-Verlag, New York, 1984.

Appendices

Appendix A

Filtering Technique

The filtering technique used in this thesis employed Weiner filtering concepts [2] to establish a cutoff frequency used in the digital filter. This idea was presented by Frankel [2], and, therefore, this discussion draws heavily from this reference. First, the discreet Fourier transform (DFT) of the data defined by

$$\hat{\Psi}(\varpi_n) = \sum_{k=0}^{N-1} \Psi(t_k) e^{\frac{-2\pi nk}{N}}, \quad n = 0,1,...,N-1 \tag{A.1}$$

with formal inversion

$$\Psi(t_k) = \frac{1}{N} \sum_{n=0}^{N-1} \hat{\Psi}(\varpi_n) e^{\frac{2\pi ink}{N}}, \quad k = 0,1,...,N-1 \tag{A.2}$$

Note that Ψ can be temperature or heat flux. In order to obtain a power spectrum, the coefficients a_n, b_n were defined as

$$a_n = \sum_{k=0}^{N-1} \Psi(t_k) \cos\left(\frac{2\pi nk}{N}\right), \quad n = 0,1,...,N-1 \tag{A.3}$$

$$b_n = \sum_{k=0}^{N-1} \Psi(t_k) \sin\left(\frac{2\pi nk}{N}\right), \quad n = 0,1,...,N-1 \tag{A.4}$$

The power density, c_n, and modified power density, C_n, were then defined as

$$c_n = \sqrt{a_n + b_n} \tag{A.5}$$

$$C_n = \frac{2}{N} c_n \tag{A.6}$$

The frequency was then defined as

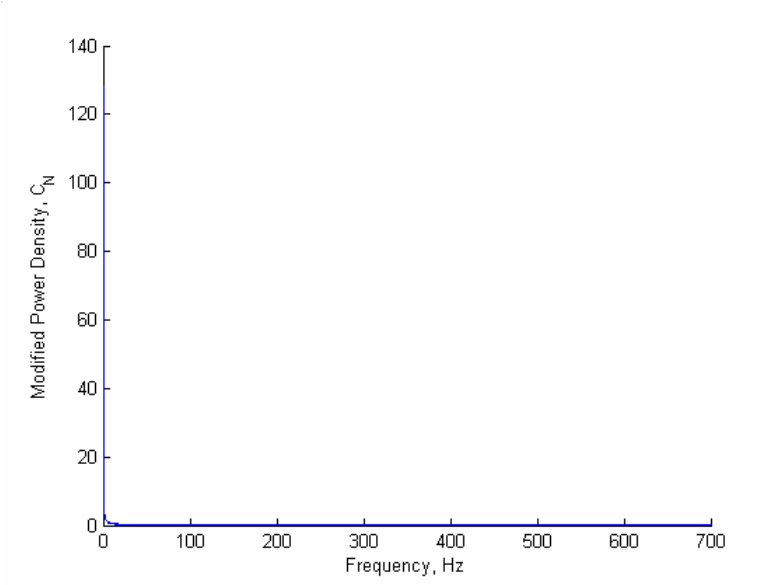
$$f = \frac{n}{t_{\max}} \tag{A.7}$$

A plot of the resulting DFT as a function of frequency can be seen in Figure (A.1). The cutoff frequency was determined as the frequency where the distance from the DFT curve to the origin was the minimum. For the example shown in Figure (A.1), a cutoff frequency of 0.5 Hz was used. This cutoff frequency, f_c, was then used the Gauss low pass filter previously defined as

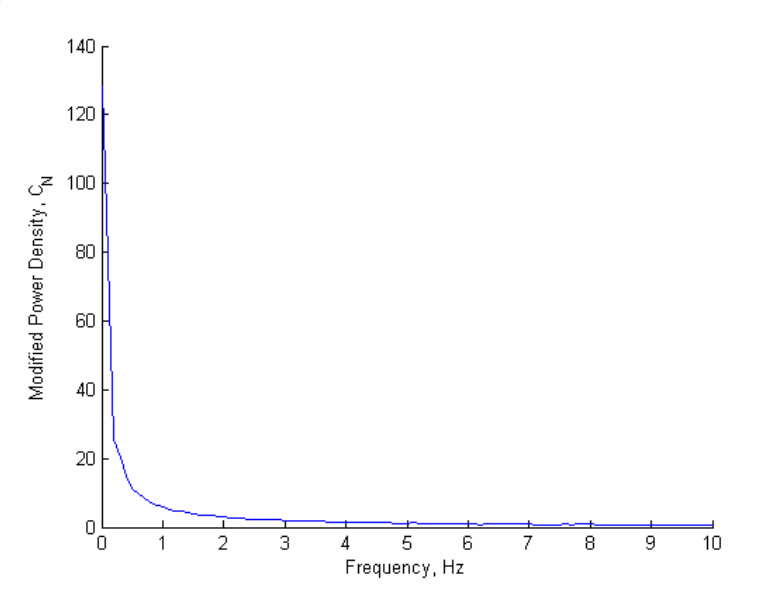
$$\Psi_{filter}(x,y,t) = \frac{\sum_{n=0}^{M} \Psi_{noise}(x,y,t_n) e^{-\frac{\left(t-t_n^2\right)\varpi_c^2}{4}}}{\sum_{n=0}^{M} e^{-\frac{\left(t-t_n^2\right)\varpi_c^2}{4}}}, \quad t \geq 0 \tag{10}$$

where $\omega_c = 2\pi f_c$. It is important to note slight change in the value of cutoff frequency has minimal effect on the resultant filtered data. Figure (A.2) shows the resultant temperature histories using $f_c = 0.5$ Hz and $f_c = 1.0$ Hz. These two temperature histories are graphically identical.

Figure (A.2) also shows the effect of the filter at the beginning and end of the history. Around t = 0, the filtered temperature history over-predicts the temperature, and near t_{max}, the filtered temperature history begins to drift downward. This is known as the filter effect, and it can be combated by padding the data with "lead data." For the inverse problem presented in this thesis, padding the data reduced the beginning filter effect to affect less than the penetration time of 0.88s.

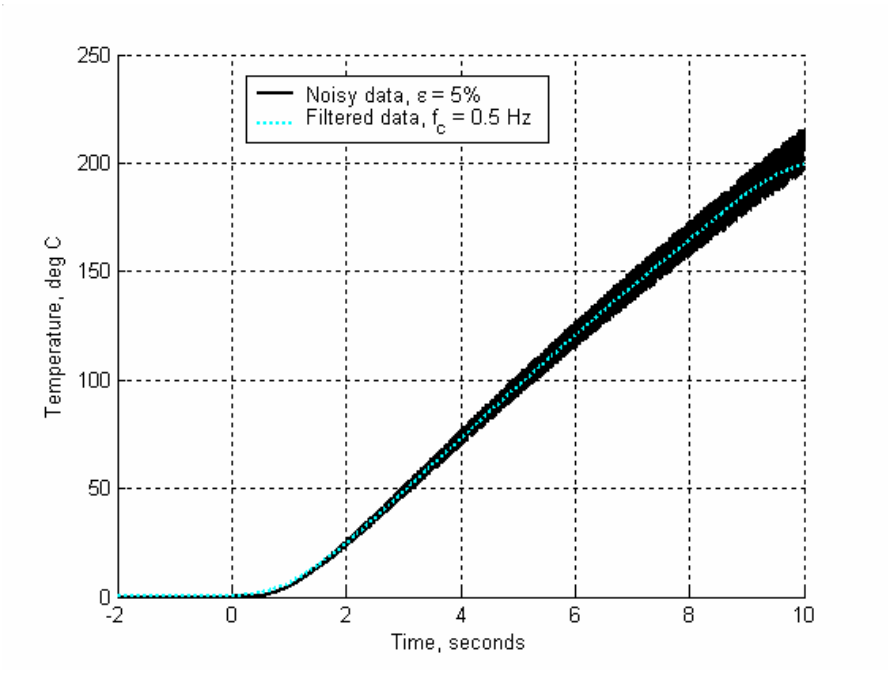


(a)

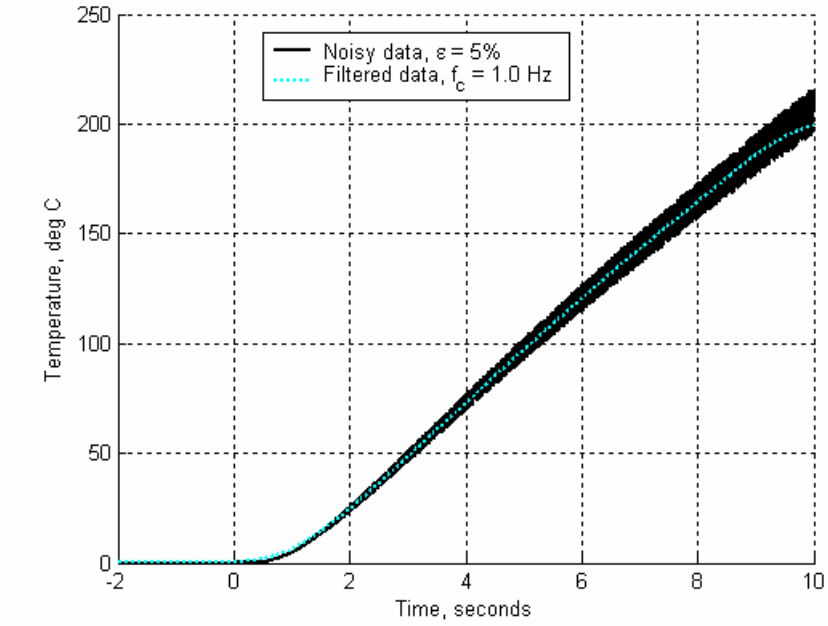


(b)

Figure A.1: Discreet Fourier Transform (DFT) of temperature data. (a) Entire spectrum and (b) zoomed in for f < 10 Hz.



(a)



(b)

Figure A.2: Effect of cutoff frequency on filtered temperature histories. (a) f_c = 0.5 Hz and (b) f_c = 1.0 Hz.

Appendix B

MATLAB codes for complete method

B.1 readme.txt file

This readme.txt file explains the overall architecture of the inverse PC simulation. All files must be saved in the same folder.

Code order:
For 1D problem,

a. direct_explicit.m, set dim=1
b. noisegenerator.m
c. filter_run.m
d. filter_postproc1D.m
e. inverse.m, set dim=1

For 2D problem,

a. direct_explicit.m, set dim=2
b. noisegenerator.m
c. filter_run.m
d. filter_ydir.m
e. filter_postproc2D.m
f. inverse.m, set dim=2

Code descriptions
1. direct_explicit.m
This code runs the direct finite difference code to generate the output necessary for the inverse problem.
Inputs:

slab properties
geometry
distance from sensor to surface

Outputs:

sensor heat flux: sensor_data_q.txt
sensor temperature: sensor_data_temp.txt
time vector: sensor_data_time.txt
y-spacing of sensors: sensor_data_y.txt
temperature exact soln: sensor_data_T_0_0.txt
y-spacing of exact soln: sensor_data_y00.txt

2. noisegenerator.m

This short program creates noise (unscaled) matrices for the temperature and heat flux data and saves it to file

Inputs:

sensor heat flux and temperature data

Outputs:

unscaled temperature noise matrix filter_noise_T.txt
unscaled heat flux noise matrix filter_noise_q.txt

3. filter_run.m

This file 1)loads sensor data and noise matrices, 2)adds noise to temperature and heat flux data, 3)filters temperature and heat flux in time, 4)exports filter temperature and heat flux histories.

Inputs:

sensor data
noise matrices
cutoff frequency

Outputs:

filtered heat flux filter_data_q.txt
filter time matrix filter_data_time.txt
filtered temperature filter_data_temp.txt

4. filter_ydir.m

This code filters the temperature and heat flux data in the y-direction after it has already been filtered in time. This is done to ensure smoothness along the y-line of sensors which in turn ensures an accurate heat flux prediction. Since a limited number of thermocouples are assumed to be available from the direct code, this code increases the number available for the integration in filter_postproc.m

Since this code filters in the y-direction, it should only be used for two-dimensional case

Inputs:

filtered heat flux and tempearture data
cutoff frequency (will be different than filter_run.m)

Outputs:

filtered temperature yfilter_data_temp.txt
filter heat flux yfilter_data_q.txt
filter y vector yfilter_data_y.txt

5. filter_postproc1D.m

This code is used for the 1D case to determine the heat flux at the sensor site using only the sensor (filtered) temperature data

Ref: [2] Frankel, J.I., "Regularization of Inverse Heat Conduction by Combination of Rate Sensor Analysis and Analytic Continuation," Journal of Engineering Mathematics, 57, 2007, pp. 181-198.

Inputs:

filtered temperature data
slab properties

Outputs:

integral heat flux filter_data_qpost.txt
all other data passes through filter_data_????post.txt

6. filter_postproc2D.m
This code is used for the 2D case to determine the heat flux at the sensor site using only the sensor (filtered) temperature data
Ref: [3] Frankel, J.I., Keyhani, M., Arimilli, R.V., and Wu, J., "A New Multidimensional Integral Relationship between Heat Flux and Temperature for Direct Internal Assessment of Heat Flux," Zeitschrift für Angewandte Mathematik und Physik, ZAMP, 2007; 58:1-20.
Inputs:

filtered temperature data
slab properties

Outputs

integral heat flux filter_data_qpost.txt
all other data passes through filter_data_????post.txt

7. inverse.m
This is the inverse code used to generate an inverse predicted temperature and heat flux solution at the surface. Data is loaded from the post processor as input to this code. A fully implicit in time approachto the FD eqns is used combined with space marching ideas.
Inputs:

post processing data
slab properties
geometry

Outputs (plots):

inverse heat flux solution
inverse temperature solution

B.2 direct_explicit.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Bryan Scott Elkins
%May 2008
%
%This is a direct finite difference code used to generate the temperature
```

```
%and/or heat flux at the sensor site for inverse analysis.  See Figure 4a
%for geometry
%
%
%
%Important variables
%dim        =   dimension of problem.  present code handles 1 or 2
dimensions
%b          =   half-width of heating element (for 1D case, this is
unimportant)
%sense_x    =   x-distance from surface to sensor (scalar)
%T0         =   initial temperature everywhere in body (scalar)
%leng       =   length (y-direction) of slab
%wide       =   width (x-direction) of slab
%M          =   number of nodes in x-direction
%N          =   number of nodes in y-direction
%tnum       =   number of time steps
%dx, dy, dt =   nodal spacing for x, y, and time, respectively
%Fo         =   nodal Fourier number, must be > 1/4 for stability
%T,q        =   temperature, heat flux matrices, respectively
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Clear all memory and start calculation timer
clear all;
clc;
tic;

%Specify 1D or 2D problem
dim = 2;    %dim=1 for 1D or 2 for 2D

T0 = 0;         %initial temperature everywhere
b = .01;        %half-width of heating element, m
q1_elem = 1e6;  %heating element heat addition per unit area (W/m^2)
sense_x = b/2;  %x-distance from surface to sensor

%Specify geometry based on dimension of problem
if dim ==1

    leng = 1*b;     %Length of surface (y-dir), m
    wide = 15*b;    %width of slab (x-dir), m
    L_heat = [-leng leng];    %endpoints of heating element
    M = 301;    %number of nodes in x-direction
    N = 3;    %number of nodes in y-direction

elseif dim==2
    leng = 15*b;    %Length of surface (y-dir), m
    wide = 7.5*b;   %width of slab (x-dir), m
    L_heat = [-b b];        %endpoints of heating element
    M = 301;    %number of nodes in x-direction
    N = 601;    %number of nodes in y-direction

end

%Slab properties - using AISI 304 Stainless Steel
rho = 7900;         %kg/m^3
k = 14.70;     %W/(m*K)
```

```
alpha = 3.75e-6;    %m^2/s
Cp = k/(alpha*rho);    %J/(kg*K)

dy = leng/(N-1);    %y-direction node spacing
dx = wide/(M-1);    %x-direction node spacing
tmax = 10;      %seconds
tnum = 10001;   %number of time steps
y = -leng/2:dy:leng/2;

dt = tmax/(tnum-1); %time step size
t = 0:dt:tmax;
Fo = alpha*dt*(1/dx^2); %nodal Fourier number

T = zeros(M,N); %initialize temperature matrix, degrees K
T_k1 = T;       %initialize temperature at previous time matrix
T(:,:,1) = T0;  %set temperature at t=0 equal to initial temperature
T_b_y_t = zeros(N,tnum);    %initialize sensor temperature matrix
q_b = zeros(N,tnum);        %initialize sensor heat flux matrix, Watts

%set boundary conditions
Teast = T0;
Tsout = T0;
Twest = T0;
T(:,1,:) = Twest;
T(:,N,:) = Teast;
T(M,:,:) = Tsout;

%Patankar's fully explicit finite difference scheme is used here.  This
%loop initializes temperature coefficients for the domain. n,s,e,w,p,p0
%represent north, south, east, west, center, center previous time
%respectively
%
%Loop also creates the source heat flux boundary condition for the north
%face of geometry.  see Figure 4a.
ap0 = zeros(M,N);
an = zeros(M,N);
as = zeros(M,N);
ae = zeros(M,N);
aw = zeros(M,N);
ap = zeros(M,N);
for ii=1:M
    for jj=1:N
        ap0(ii,jj) = rho*Cp*dx*dy/dt;
        ae(ii,jj) = k*dx/dy;
        aw(ii,jj) = k*dx/dy;
        as(ii,jj) = k*dy/dx;

        if ii==1
            %create source heat flux boundary condition
            if y(jj) < L_heat(1) | y(jj) > L_heat(2)
                qn(jj) = 0;
            else
                qn(jj) = q1_elem*dy;
            end
            ap0(ii,jj) = ap0(ii,jj)/2;
            ae(ii,jj) = ae(ii,jj)/2;
            aw(ii,jj) = aw(ii,jj)/2;
            ap(ii,jj) = ap0(ii,jj);
```

```
        else
            an(ii,jj) = k*dy/dx;
            ap(ii,jj) = ap0(ii,jj);
        end
    end
end
ap0 = ap0 - (an + as + ae + aw);



%main finite difference loop
%solve for temperature distribution throughout domain and calculate heat
%flux at sensor location
%
%solve for entire spatial domain at one time step, then move to next time
%step
for tt=2:tnum   %march in time
    fprintf('\n%7.4f',t(tt)) %print current time to command window

    %set temperature at previous time equal to previous time matrix
    T_k1 = T;
    Tnew = T_k1;
    b1 = ap0.*T_k1; %Patankar's source term

    for ii=1:M      %march in x-direction
        for jj=1:N  %march in y-direction

            %Find neighboring temperatures
            if jj==1
                Tw = T_k1(ii,jj+1);
            else
                Tw = T_k1(ii,jj-1);
            end
            if jj==N
                Te = T_k1(ii,jj-1);
            else
                Te = T_k1(ii,jj+1);
            end
            if ii==M
                Ts = T_k1(ii-1,jj);
            else
                Ts = T_k1(ii+1,jj);
            end

%Solve temperature at current note
%Store exact solution if at surface
%Store temperature and heat flux if at sensor
            if ii==1
                Tnew(ii,jj) = (ae(ii,jj)*Te + aw(ii,jj)*Tw + as(ii,jj)*Ts +
qn(jj) + b1(ii,jj))/ap(ii,jj);
                if jj==(N+1)/2+1
                    T_0_0_t(tt) = Tnew(ii,jj);
                end
            else
                Tn = T_k1(ii-1,jj);
                Tnew(ii,jj) = (ae(ii,jj)*Te + aw(ii,jj)*Tw + an(ii,jj)*Tn +
as(ii,jj)*Ts + b1(ii,jj))/ap(ii,jj);
                if ii==round(sense_x/dx)+1
```

```
                    T_b_y_t(jj,tt) = Tnew(ii,jj);
                    q_b(jj,tt) = -((ae(ii,jj) + aw(ii,jj) + an(ii,jj) +
ap0(ii,jj))*Tnew(ii,jj) -...
                        (ae(ii,jj)*Te + aw(ii,jj)*Tw + as(ii,jj)*Ts +
b1(ii,jj)));
                end
            end
        end
    end
    T = Tnew;
end
%end loop

%Print processing time and direct method results to command window
toc
fprintf('\n max surface temp = %14.3f',T_0_0_t(tnum))
fprintf('\n max sensor temp  = %14.3f',T_b_y_t((N+1)/2,tnum))
fprintf('\n Fourier number   = %14.5f',Fo)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%pull out the sensors if 2D
if dim == 2
    dy_s = 0.005;       %maximum physical distance possible between sensors
    skip_y = dy_s/dy;   %number of nodes that need to be skipped

    ys = -leng/2:dy_s:leng/2;
    Ns = length(ys);
    T_sens = T_b_y_t(:,:);
    q11_sens = q_b./dy;

%simple routine to pull out sensors info, maintaining the same amount of
%heat flux from the sensors
    pp = 0;
    for nn = 1:skip_y:N
        pp = pp+1;
        temp = 0;
        for rr=1:skip_y/2-1
            if nn==1
            else
                temp = temp + q_b(nn-rr,:);
            end
            if nn==N
            else
                temp = temp + q_b(nn+rr,:);
            end
        end
        q_sense(pp,:) = q_b(nn,:) + temp;
        if nn~=1
            q_sense(pp,:) = q_sense(pp,:) + q_b(nn-skip_y/2,:)/2;
        end
        if nn~=N
            q_sense(pp,:) = q_sense(pp,:) + q_b(nn+skip_y/2,:)./2;
        end
        T_sense(pp,:) = T_sens(nn,:);
    end
elseif dim==1
    T_sense = T__b_y_t;
```

```
    q_sense = q_b;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Save sensor and exact solution data to files for use in inverse code
save sensor_data_q.txt q_sense -ascii -double
save sensor_data_time.txt t -ascii -double
save sensor_data_temp.txt T_sense -ascii -double
save sensor_data_y.txt ys -ascii -double
save sensor_data_y00.txt y -ascii -double
save sensor_data_T_0_0.txt T_0_0_t -ascii –double
%end of program
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

B.3 noisegenerator.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This short program creates noise (unscaled) matrices for the temperature
%and heat flux data and saves it to file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%clear memory
clear all;
clc;

%load temperature and heat flux data (for size)
qs = load('sensor_data_q.txt');
Ts = load('sensor_data_temp.txt');

%generate random (uniform) noise matrices
noise = rand(size(Ts));
noiseq = rand(size(qs));

%save noise matrices to files
save filter_noise_T.txt noise -ascii -double
save filter_noise_q.txt noiseq -ascii -double

%end program
```

B.4 filter_run.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This file 1)loads sensor data and noise matrices, 2)adds noise
%temperature and heat flux data, 3)filters temperature and heat
%flux in time, 4)exports filter temperature and heat flux histories
%
%Key variables
%Ts,qs      =   sensor temperature and heat flux matrices, respectively
```

```
%npercent   =   noise level (in percent of exact temperature)
%tpad       =   length of lead "pad" time to add
%fc         =   cutoff frequency, Hz
%tnumnew    =   number of time nodes for filtered data (including padding)
%Tfilter    =   filtered temperature
%qfilter    =   filtered heat flux
%
%Note that the exact surface temperature, T00, is also filtered (with a
%high cutoff frequency to ensure no loss of data).  This is only to enable
%a temperature history at the same point in time as the inverse solution.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Clear memory
clear all;
clc;
tic

%load input data from direct code and noise generator
qs = load('sensor_data_q.txt');
ts = load('sensor_data_time.txt');
Ts = load('sensor_data_temp.txt');
ys = load('sensor_data_y.txt');
T00 = load('sensor_data_T_0_0.txt');
noise = load('filter_noise_T.txt');
noiseq = load('filter_noise_q.txt');

%define noise level using T = T*(1 + e) method
npercent = 5/100;
nlevel = npercent.*Tnoise;
noise = noise.*2.*nlevel - nlevel;
Tnoise = Ts + noise;
nlevelq = npercent.*qnoise;
noiseq = noiseq.*2.*nlevelq - nlevelq;
qnoise = qs + noiseq;

%read time/space information from input data
tmax = max(ts);
tnum = length(ts);
dt = ts(2)-ts(1);
N = length(ys);
dy = ys(2)-ys(2);
Tsmax = max(max(abs(Ts)));

%pad data with initial value to yield a smoother filtered result
tpad = 2;
textra = -tpad:dt:0;
tpadnum = length(textra);
t = ts;
t(tpadnum:tpadnum+tnum-1) = t;
t(1:tpadnum)=textra;
ts = t;
qs(:,tpadnum:tpadnum+tnum-1) = qs;
qs(:,1:tpadnum)=0;
Ts(:,tpadnum:tpadnum+tnum-1) = Ts;
Ts(:,1:tpadnum)=0;
T00(tpadnum:tpadnum+tnum-1) = T00;
T00(1:tpadnum)=0;
tnum = length(ts);
```

```
Tnoise(:,1:tpadnum) = 0;
qnoise(:,1:tpadnum) = 0;

%define cutoff frequency
fc = .5;        %Hz
wc = 2*pi*fc;   %rad/s
fcq = 1;        %currently not using
wcq = 2*pi*fcq; %not currently using

%find temp/flux at center node - used for plots at bottom
Tc = Ts((N+1)/2,:);
Tcnoise = Tnoise((N+1)/2,:);
qc = qs((N+1)/2,:);
qcnoise = qnoise((N+1)/2,:);

%Define number of time nodes for filtered result
tnumnew = 301;
dtnew = 12/(tnumnew-1);
tnew = -tpad:dtnew:10;
Tfilter = zeros(N,tnumnew);
qfilter = zeros(N,tnumnew);
T00filter = zeros(1,tnumnew);
numtemp = zeros(tnum,tnumnew);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%actual filter begins here

%first loop finds the denominator and half the numerator
for k=0:tnumnew-1
    numtemp(:,k+1) = exp(-(tnew(k+1)-t).^2.*wc.^2./4)';
    den(k+1) = sum(exp(-(tnew(k+1)-t).^2.*wc.^2./4));
end

%second loop find second of the numerator and the filtered results
for jj=1:N
    num = Tnoise(jj,:)*numtemp;
    numq = qnoise(jj,:)*numtemp;
    Tfilter(jj,:) = num./den;
    qfilter(jj,:) = numq./den;
end

%run filter again for exact surface temperature
fc00 = 400;     %note the high cutoff frequency, 400 Hz
wc = 2*pi*fc00;
for k=0:tnumnew-1
    numtemp(:,k+1) = exp(-(tnew(k+1)-t).^2.*wc.^2./4)';
    den(k+1) = sum(exp(-(tnew(k+1)-t).^2.*wc.^2./4));
end
numT00 = T00*numtemp;
T00filter = numT00./den;
toc

%end of filter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%save filtered data to file
save filter_data_q.txt qfilter -ascii -double
save filter_data_time.txt tnew -ascii -double
```

```
save filter_data_temp.txt Tfilter -ascii -double
save filter_data_y.txt ys -ascii -double
save filter_data_T_0_0.txt T00filter -ascii -double

%end of program
```

B.5 filter_ydir.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This code filters the temperature and heat flux data in the y-direction
%after it has already been filtered in time.  This is done to ensure
%smoothness along the y-line of sensors which in turn ensures an accurate
%heat flux prediction.  Since a limited number of thermocouples are assumed
%to be available from the direct code, this code increases the number
%available for the integration in filter_postproc.m
%
%Since this code filters in the y-direction, it should only be used for
%two-dimensional case

%Key variables
%Ts,qs      =   sensor temperature and heat flux matrices, respectively
%fc         =   cutoff frequency, Hz
%Nnew       =   number of nodes in y-direction for filtered data
%Tfilter    =   filtered temperature
%qfilter    =   filtered heat flux
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%clear memory
clear all;
clc;
tic

%load input data from time filter
ts = load('filter_data_time.txt');
Ts = load('filter_data_temp.txt');
qs = load('filter_data_q.txt');
ys = load('filter_data_y.txt');
T00 = load('filter_data_T_0_0.txt');

%define "noisy" matrices and intrepret information from filter data
Tnoise = Ts;
qnoise = qs;
tmax = max(ts);
tnum = length(ts);
dt = ts(2)-ts(1);
N = length(ys);     %number of y-nodes
dy = ys(2)-ys(2);
Tsmax = max(max(abs(Ts)));  %maximum sensor temperature

%define cutoff frequency
fc = 100;       %Hz
wc = 2*pi*fc;   %rad/s
```

```
%define new y vector
Nnew = 301;     %number of y nodes
y=ys;
Y = max(y)-min(y);
dynew = Y/(Nnew-1);
ynew = min(y):dynew:max(y);

%initialize filter matrices
Tfilter = zeros(Nnew,tnum);
T00filter = T00;
numtemp = zeros(Nnew,N);
numtempq = zeros(Nnew,N);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%actual filter begins here

%first loop finds the denominator and half the numerator
for k=0:Nnew-1
    numtemp(k+1,:) = exp(-(ynew(k+1)-y).^2.*wc.^2./4);
    den(k+1) = sum(exp(-(ynew(k+1)-y).^2.*wc.^2./4));
end

%second loop find second of the numerator and the filtered results
for jj=1:tnum
    num = Tnoise(:,jj)'*numtemp';
    Tfilter(:,jj) = (num./den)';
    numq = qnoise(:,jj)'*numtemp';
    qfilter(:,jj) = (numq./den)';
end

toc
%end of filter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%save filtered data to file
save yfilter_data_time.txt ts -ascii -double
save yfilter_data_temp.txt Tfilter -ascii -double
save yfilter_data_q.txt qfilter -ascii -double
save yfilter_data_y.txt ynew -ascii -double
save yfilter_data_T_0_0.txt T00filter -ascii -double

%end of program
```

B.6 filter_postproc1D.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This code is used for the 1D case to determine the heat flux at the
%sensor site using only the sensor (filtered) temperature data
%
%Ref: [2]    Frankel, J.I., "Regularization of Inverse Heat Conduction by
%Combination of Rate Sensor Analysis and Analytic Continuation,"
%Journal of Engineering Mathematics, 57, 2007, pp. 181-198.
%
```

```
%Key Variables
%T      =   filtered sensor temperatures
%T1     =   heating rate, dT/dt
%T2     =   d^2T/dt^2
%t      =   time
%u      =   dummy time variable
%q11    =   integral heat flux, W/m^2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Clear memory
clear all;
clc;

%Load filter data
Tfil = load('filter_data_temp.txt');
qfil = load('filter_data_q.txt');
t = load('filter_data_time.txt');
y = load('filter_data_y.txt');
T00s = load('filter_data_T_0_0.txt');

%interrogate data to get mesh info
T = Tfil;
tmin = min(t);
t = t-tmin;     %subtract padding time so time starts at 0
tnum = length(t);
dt = t(2) - t(1);
dy = y(2) - y(1);
N = length(y);

%Slab properties - using AISI 304 Stainless Steel
rho = 7900;     %kg/m^3
k_univ = 14.70;        %W/(m*K)
alpha = 3.75e-6;        %m^2/s
Cp = k_univ/(alpha*rho);
k=k_univ;

%Forward Difference for heating rate
for ii=1:tnum-1
    T1(:,ii) = (T(:,ii+1) - T(:,ii))./dt;
end
T1(:,tnum) = T1(:,tnum-1);

%forward difference for derivative of heating rate
for ii=1:tnum-1
    T2(:,ii) = (T1(:,ii+1) - T1(:,ii))./dt;
end
T2(:,tnum) = T2(:,tnum-1);

K = (rho*Cp*k/pi)^.5;   %constant to multiply integral with

%begin integral loop
for tt=1:length(t)
    fprintf('\n%2.2f',t(tt))    %print progress
    u = t(1):dt:t(tt);          %dummy time variable
    ii = 1:length(u);
    U = length(u);

    %find heat flux
```

```
    for jj=1:N
        temp = T2(jj,ii).*(t(tt)-u).^.5;
        q11(jj,tt) = K*dt/3*2 * (temp(1) + 4*sum(temp(2:2:U)) +
2*sum(temp(3:2:U)) + temp(U));
    end
end

q = -q11.*dy;   %heat flux, W
t = t+tmin;     %restore time padding

%Save heat flux data to disk
save filter_data_qpost.txt q11 -ascii -double
save filter_data_tpost.txt t -ascii -double
save filter_data_ypost.txt y -ascii -double
save filter_data_temppost.txt T -ascii -double
save filter_data_T00post.txt T00s -ascii -double

%end of program
```

B.7 filter_postproc2D.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This code is used for the 2D case to determine the heat flux at the
%sensor site using only the sensor (filtered) temperature data
%
%Ref: [3]    Frankel, J.I., Keyhani, M., Arimilli, R.V., and Wu, J.,
%"A New Multidimensional Integral Relationship between Heat Flux and
%Temperature for Direct Internal Assessment of Heat Flux," Zeitschrift
%für Angewandte Mathematik und Physik, ZAMP, 2007; 58:1-20.
%
%Key Variables
%T      =   filtered sensor temperatures
%T1     =   heating rate, dT/dt
%t      =   time
%t0     =   dummy time variable
%y      =   spatial variable
%y0     =   dummy spatial variable
%q11    =   integral determined heat flux, W/m^2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%clear memory
clear all;
clc;

%load sensor data from y-direction filter
qfila = load('yfilter_data_q.txt');
ta = load('yfilter_data_time.txt');
Tfila = load('yfilter_data_temp.txt');
ya = load('yfilter_data_y.txt');
T00s1 = load('yfilter_data_T_0_0.txt');

%specify if all computational nodes should be kept
skipt = 1;      %number of time nodes to skip
```

```
skipy = 1;      %number of y nodes to skip

%find new time, temperature, and flux data with appropriate skipping
t = ta(1:skipt:length(ta));
Tfilb = Tfila(:,1:skipt:length(ta));
qfilb = qfila(:,1:skipt:length(ta));
T00s = T00s1(1:skipt:length(ta));
y = ya(1:skipy:length(ya));
Tfil = Tfilb(1:skipy:length(ya),:);
qfil = qfilb(1:skipy:length(ya),:);
T = Tfil;

%interrogate data to obtain mesh properties
tmin = min(t);
tnum = length(t);
N = length(y);
dt = t(2) - t(1);
dy = y(2) - y(1);

%Slab properties - using AISI 304 Stainless Steel
rho = 7900;     %kg/m^3
k_univ = 14.70;        %W/(m*K)
alpha = 3.75e-6;        %m^2/s
Cp = k_univ/(alpha*rho);
k=k_univ;


%obtain heating rate via forward difference of temperature data
T1 = (T(:,2:tnum) - T(:,1:tnum-1))./dt;
T1(:,tnum) = 2*T1(:,tnum-1)-T1(:,tnum-2);

%simple numerical trick to avoid t = t0 to cause singularity issues in trap
%method
t0 = t-dt/1000000;

%Loop to evaluate heat flux integral, eqn 17 in Frankel's ZAMP paper
%space marching
for mm=1:length(y)
    fprintf('\n%2.2f',mm/length(y)*100)
    %time marching
    for ii=1:length(t)

        %first term in integration
        term1(mm,ii) = 2*k/((alpha*pi)^.5)*(t(ii)^.5)*T1(mm,ii);
        temp1 = zeros(1,ii);
        for jj = 1:ii
            term2 = zeros(1,length(y));
            for nn=1:length(y)

                %evaluate integrand one piece at a time
                term2a = exp(-((y(mm)-y(nn))^2)/(4*alpha*(t(ii)-
t0(jj)))).*(t(ii)-t0(jj));
                term2b = T1(nn,jj) - T1(mm,ii);
                term2c = 1/(2*(t(ii)-t0(jj))) * (1 - (y(mm)-y(nn))^2 /
(2*alpha*(t(ii)-t0(jj))));
                term2d = T(nn,jj) - T(mm,ii);
                term2(nn) = term2a*(term2b + term2c*term2d);
            end
```

```
            %integrate over y
            temp1(jj) = .5*dy*( 2*sum(term2) - term2(1) - term2(length(y))
);%trap
            %temp1(jj) = dy/3* ( term2(1) + 4*sum(term2(2:2:N)) +
2*sum(term2(3:2:N)) + term2(N) );%simpsons rule
        end

        %integrate over time
        temp2(mm,ii) = .5*dt*( 2*sum(temp1) - temp1(1) - temp1(ii) );%trap
        %temp2 = dt/3* ( temp1(1) + 4*sum(temp1(2:2:ii)) +
2*sum(temp1(3:2:ii)) + temp1(ii) );%simpsons rule

        %put all pieces together for q11, W/m^2
        q11(mm,ii) = term1(mm,ii) + k/(2*alpha*pi).*temp2(mm,ii);
    end
end

%save results to file
save filter_data_qpost.txt q11 -ascii -double
save filter_data_tpost.txt t -ascii -double
save filter_data_ypost.txt y -ascii -double
save filter_data_temppost.txt T -ascii -double
save filter_data_T00post.txt T00s -ascii -double

%end program
```

B.8 inverse.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This is the inverse code used to generate an inverse predicted temperature
%and heat flux solution at the surface.  Data is loaded from the post
%processor as input to this code.  A fully implicit in time approachto the
%FD eqns is used combined with space marching ideas.
%
%key variables
%dim            =   dimesion of problem, each dim had a different geometry
%q_sense        =   filtered, sensor heat flux data
%T_sense        =   filtered, sensor temperature data
%M,N,tnum       =   number of nodes in x,y,time respectively
%skip_y,skip_t  =   number of nodes to skip from filter in y,time resp.
%sense_x        =   x-distance from sensor to surface
%Tsurf      =   inverse solution surface temperature
%qn         =   inverse solution surface heat flux
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%clear memory
clear all;
clc;
tic;

%load filter and post processing data
```

```
%the call below uses the integral determined heat flux
q_sense1 = load('filter_data_qpost.txt');
t_sense1 = load('filter_data_tpost.txt');
T_sense1 = load('filter_data_temppost.txt');
y_sense1 = load('filter_data_ypost.txt');
T00s1 = load('filter_data_T00post.txt');

%ensure any padded data is, in fact, equal to the initial condition
tpad = abs(min(t_sense1));
tmax = max(t_sense1);
tnum = length(t_sense1);;
tpadnum = round(tpad/(tmax + tpad)*(tnum-1))+1;
T_sense1(:,1:tpadnum)=0;
q_sense1(:,1:tpadnum)=0;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%User input section
M = 100;        %number of x-nodes to use
skip_y = 10;    %number of y-nodes to skip from filter (matches exact data)
skip_t = 1;     %40;    %number of time steps to skip from filter
dim = 2;        %dimension of inverse problem, use 2 if unknown, 1=1D, 2=2D

sense_x = .005; %distance from sensor to surface

lcolor = 'k';   %plot line color
lspec = 'k:';   %plot line spec detail
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%reform time, y, temperature, and heat flux matrices after skipping
t_sense = t_sense1(1:skip_t:length(t_sense1));
T_sensea = T_sense1(:,1:skip_t:length(t_sense1));
q_sensea = q_sense1(:,1:skip_t:length(t_sense1));
T00s = T00s1(1:skip_t:length(t_sense1));
y_sense = y_sense1(1:skip_y:length(y_sense1));
T_sense = T_sensea(1:skip_y:length(y_sense1),:);
q_sense = q_sensea(1:skip_y:length(y_sense1),:);

%knowns used for geometry and exact solution comparison, based on dimension
Ti = T_sense(1,1);
b = .01;            %half width of heating element
wide = sense_x;     %width of slab (x-dir), m
if dim==1
    leng = 1*b;    %Length of surface (y-dir), m
    y_heat = [-15*b 15*b];  %width matrix for heating element
elseif dim==2
    leng = 15*b;        %Length of surface (y-dir), m
    y_heat = [-b b];    %width matrix for heating element
end

%interrogate data to obtain mesh parameters
N = length(y_sense);        %number of y-nodes
tmax = max(t_sense);        %max time, seconds
tnum = length(t_sense);     %number of time steps
dt = t_sense(2)-t_sense(1); %time step size
tplot = round(tnum - 2/dt); %a good time to plot at
                            %(2 seconds before end to avoid filter effects
dy = leng/(N-1);    %y-spacing
```

```
dx = wide/(M-.5);   %x-spacing

%Slab properties - using AISI 304 Stainless Steel
rho = 7900;     %kg/m^3
k_univ = 14.70;        %W/(m*K)
alpha = 3.75e-6;        %m^2/s
Cp = k_univ/(alpha*rho);
q11_elem = 1e6;
k = k_univ;  %initialize conductivity matrix

%initialize temperature and heat flux matrices
T = zeros(M,N);     %initialize temperature matrix (x,y)
T_k1 = T;       `   %temp at previous time matrix   (x,y)
Tsurf = zeros(N,tnum);      %surface temp matrix (x,t)
q_sense = q_sense *(-1)*dy; %convert q11 (W/m^2) from sensor to W/m
qn = zeros(N,tnum);     %flux at surface
qn_exact = zeros(1,N);  %exact flux for comparison

%temperature coefficients using Patankar's fully implicit scheme
ae = (k*dx/dy);
aw = ae;
an = (k*dy/dx);
as = an;
ap0 = (rho*Cp*dx*dy/dt);
ap = ae + aw + as + an + ap0;

%coefficients for the surface nodes - half CV and no north term (use flux)
aestarstar = ae/2;
awstarstar = aw/2;
ap0starstar = ap0/2;
apstarstar = aestarstar + awstarstar + as + ap0starstar;

%coefficient for sensor site - use qsensor instead of Tsouth
apstar = ap - as;

%generate exact surface flux for comparison
for jj=1:N
    if y_sense(jj) < y_heat(1) | y_sense(jj) > y_heat(2)
        qn_exact(jj) = 0;
    else
        qn_exact(jj) = q11_elem;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%begin inverse code

%loop
t = t_sense;
%time marching
for tt = 2:tnum
    T_k1 = T;
    fprintf('\n%7.4f',t(tt))
    Tnew = T;

%space marching in x-dir
    for ii=M+1:-1:1
```

```
%space marching in y-dir
        for jj = 1:N

            %special case of at sensor site
            if ii==M+1

                %define neighboring temperatures
                Tp = T_sense(jj,tt);
                if jj==N
                    Te = T_sense(jj,tt);
                else
                    Te = T_sense(jj+1,tt);
                end
                if jj==1
                    Tw = T_sense(jj,tt);
                else
                    Tw = T_sense(jj-1,tt);
                end

                %solve temperature at one node north
                T(ii-1,jj) = 1/an*(apstar*Tp - ae*Te - aw*Tw -
ap0*T_sense(jj,tt) - q_sense(jj,tt)) ;
            else

            %all other nodes
                %define neighboring temps
                if jj==1
                    Tw = T(ii,jj+1);
                else
                    Tw = T(ii,jj-1);
                end
                if jj==N
                    Te = T(ii,jj-1);
                else
                    Te = T(ii,jj+1);
                end
                Tp = T(ii,jj);

                %solve based on geometry location
                %one line above sensor
                if ii==M
                    Ts = T_sense(jj,tt);
                    T(ii-1,jj) = 1/an*(ap*Tp - ae*Te - aw*Tw -
ap0*T_k1(ii,jj) - as*Ts) ;

                %all other nodes except surface
                elseif ii~=1
                    Ts = T(ii+1,jj);
                    T(ii-1,jj) = 1/an*(ap*Tp - ae*Te - aw*Tw -
ap0*T_k1(ii,jj) - as*Ts) ;

                %surface temp already found at previous x-location
                %solve for surface heat flux
                elseif ii==1
                    Ts = T(ii+1,jj);
                    qn(jj,tt) = (apstarstar*Tp - aestarstar*Te -
awstarstar*Tw -...
                        ap0starstar*T_k1(ii,jj) - as*Ts)/dy ;
```

```
                    Tsurf(jj,tt) = T(ii,jj);
                end
            end
        end
    end
end
%end loop

%end inverse code
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%figures
%define nodal Fourier number and center three nodes
Fo = alpha*dt/(dx^2)
y1 = (N+1)/2-1;
y2 = (N+1)/2;
y3 = (N+1)/2+1;
Tcenter = Tsurf(y2,:);

%plot surface heat flux distribution at tplot
y = -leng/2:dy:leng/2;
figure (1)
clf reset;
hold on;
plot(y/b,qn_exact,'k',y/b,qn(:,tplot),'k^','MarkerSize', 6)
ylabel('q_x^{''''}, W/m^2')
xlabel('y position, y/b ')
legend('Exact','Inverse')
axis([-8,8,-.2e6,1.2e6])


%plot temperature histories with exact solution, inverse, and sensor data
tp = 0.88;  %penetration time known from constant heat flux case in
Incropera
figure (2)
clf reset;
hold on;
plot(t_sense,T00s,'k',t-.88,Tsurf((N+1)/2,:),'k:',t,T_sense((N+1)/2,:),'k--
')
axis([0,tmax,0,500])
xlabel('time - seconds')
ylabel('Temperature, deg C')
legend('T_{direct}(0,0,t)','T_{inverse}(0,0,t)','T_{sensor}(b,0,t)',2)


qn_error = -(qn_exact' - qn(:,tplot))./q11_elem*100;    %surface flux error
qexact = ones(1,tnum).*1e6;

%plot heat flux error history
qn_error_0_0 = -(q11_elem - qn((N+1)/2,:))./q11_elem .* 100;
figure (8)
clf reset;
hold on;
plot(t,qn_error_0_0,lspec)
axis([0,10,-5,5])
ylabel('q'''' error, (%)')
xlabel('Time, seconds')
```

```
%find temperature error (adjusted for penetration time)
tp_tt = round(tp/dt);
Tav = max(Tsurf);
qav = (qn(y1,:) + qn(y2,:) + qn(y3,:))./3;
tnump = tnum-tp_tt;
Terror = (Tav((1+tp_tt):tnum) - T00s(1:tnump))./T00s(1:tnump).*100;

%plot temperature error history (adjust for penetration time)
figure (9)
clf reset;
plot(t(1:tnump),Terror,lspec)
hold on;
axis([0,10,-5,5])
ylabel('T error, (%)')
xlabel('Time, seconds')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%55
%plot 2-axis figure temperature on left, heat flux on right, time on x-axis
figure (13)
clf reset;
[ax1,h1,h2] = plotyy(t,T00s,[-2,0,10],[1e6,1e6,1e6]);           %plot temp
hold on;
[ax2,h3,h4] = plotyy(t(1:tnump),Tsurf(y2,1+tp_tt:tnum),t,qav);  %plot flux
hold off;

%even up tick marks on right and left y-axes
set(ax1(1),'XLim',[0,10],'YLim',[0,500])
set(ax1(2),'XLim',[0,10],'YLim',[.5e6,1.5e6])
set(ax2(1),'XLim',[0,10],'YLim',[0,500])
set(ax2(2),'XLim',[0,10],'YLim',[.5e6,1.5e6])
set(ax1(1),'YTick',[0,125,250,375,500])
set(ax1(2),'YTick',[.5e6,.75e6,1e6,1.25e6,1.5e6])
set(ax2(1),'YTick',[])
set(ax2(2),'YTick',[])

%set linespec properties
set(h1,'LineStyle','-','Color','k')
set(h2,'LineStyle','-','Color','k')
set(h3,'LineStyle',':','Color','k')
set(h4,'LineStyle',':','Color','k')
set(ax1(1),'YColor','k')
set(ax1(2),'YColor','k')
set(ax2(1),'YColor','k')
set(ax2(2),'YColor','k')

%label axes and legend
legend([h1,h3],'Exact','Complete Inverse, \epsilon = 5%','\Deltat =
0.04s','\Deltax = 0.0503mm')
set(get(ax1(2),'Ylabel'),'String','Heat flux, W/m^2')
set(get(ax1(1),'Ylabel'),'String','Temperature, deg C')
xlabel('Time, seconds')


%print results to command window
qe_av = mean(qn_error)*100;
qe_st = std(qn_error)*100;
Toff = Tsurf((N+1)/2,tplot) - T00s(tplot);
Toff = Toff/T00s(tplot)*100;
```

```
fprintf('\n T offset @ 8 = %7.4g',Toff)
fprintf('\n q_av error   = %7.4g',qe_av)
fprintf('\n q error std  = %7.4g',qe_st)

                                %end program
```

Vita

Bryan Elkins was born in Columbus, Ohio on June 4, 1984. He attended Northside Christian School in St. Petersburg, FL from first through the ninth grade. He then attended Mt. Juliet High School in Mt. Juliet, TN where he graduated high school in May, 2002. He began his undergraduate career at The University of Tennessee, Knoxville in August, 2002 and received a Bachelor of Science degree with a major in Aerospace Engineering in May, 2006. He began his graduate career in August, 2006 also at The University of Tennessee, Knoxville and received his Master of Science degree in May, 2008. He is happily married to the former Sara Smith of Nashville, TN.